1991年1月1日発行（毎月1回1日発行）第10巻1号通巻105号　昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 急接近! SX-WINDOW

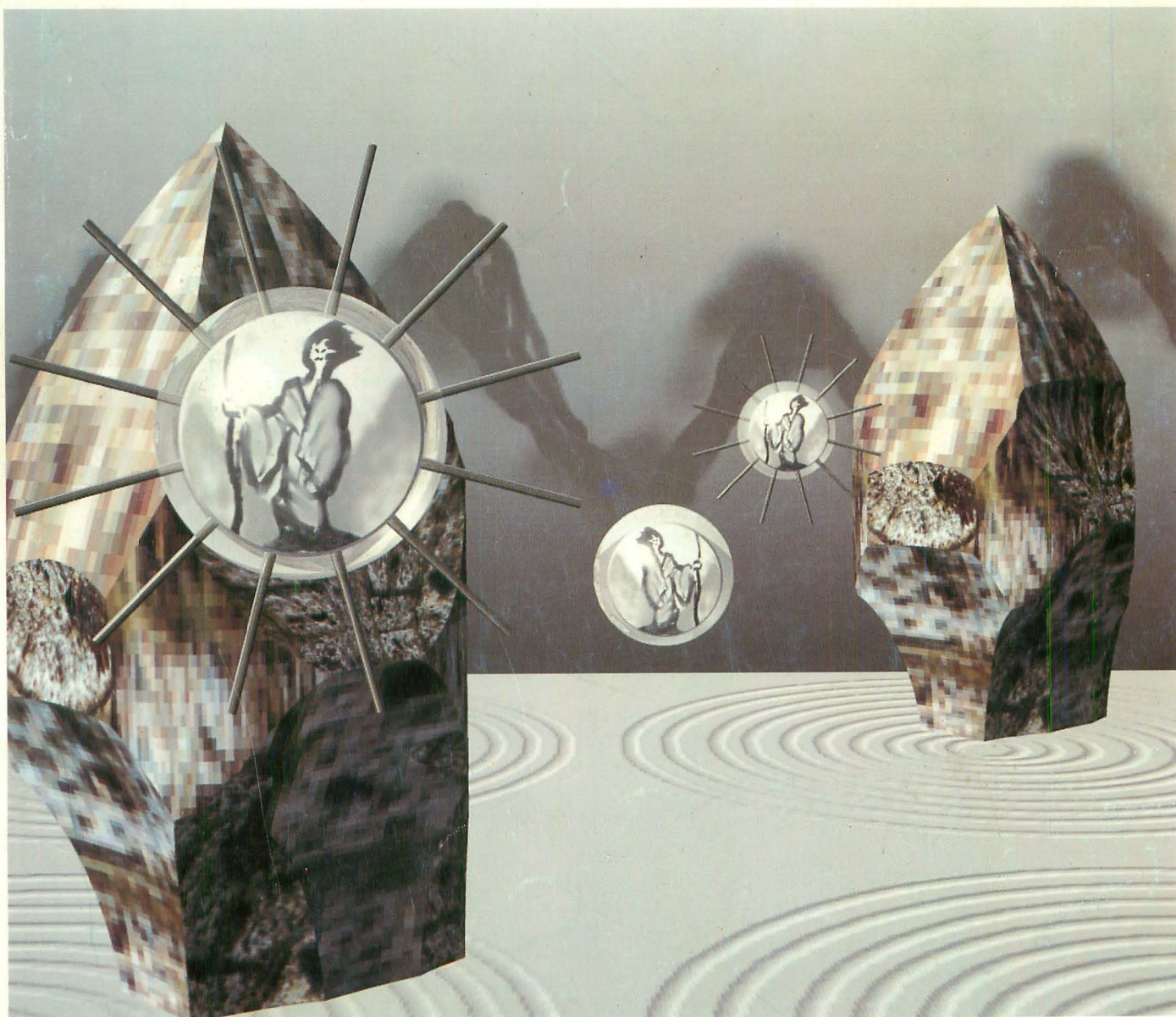
1

1991

付録5"2HDディスク
謹賀新年PRO-68K

ビジュアルシェルの拡張 VS2.X/グラフ作成ツールMichael
半影つきレイトレHASH.X/Z'sSTAFF拡張機能/カード型DB
MUSICDRV.XV.2.0/CARDDRV.X&カードゲームKLONDIKE
SX-WINDOW開発資料/ツール/15パズル/LIFE GAME他

オー!エックス
特別定価780円

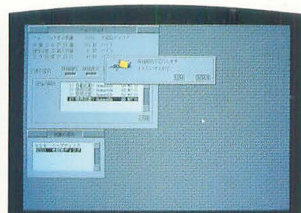
SHARP

X68000 SUPER登場

このたび新たにラインアップされた"X68000 SUPER"は、すでに発売されている"SUPER HD"と同様、SCSIインターフェイスを標準装備しています。また、その他のシリーズにはオプションとしてSCSIボード(CZ-6BS1)がサポートされ、大容量外部記憶装置をはじめ、各種SCSI装置との接続が可能になったのは、ご存じのとおりです。

SCSI規格とは……

SCSIは1986年にANSI(米国規格協会)で規格化された仕様で、Small Computer System Interfaceの略。小型コンピュータ

▲MOディスクドライブ情報

▲MOドライブディレクトリ情報

▲MOディスクのフォーマット

の周辺機器接続のための世界共通の規格です。大容量外部記憶装置(大容量ハードディスク、CD-ROM、DATなど)に加え、登場が期待される高速スキャナ、次世代プリンタなどのSCSI装置を、デイジーチェーン方式で最大7台まで接続可能。大容量データの高速転送、および単一のインターフェイスでの周辺機器の複数制御が特長です。

X68000と大容量メディア

サウンドクリエーション&コンピュータグラフィックス。X68000のオハコともいうべきこの領域は、感性あふれるユーザーにとって最も魅力的である反面、表現の繊細さに比例して必要な外部記憶容量も増大します。サンプリング、MIDI、レイトレ……。その潜在能力をフルに引き出すには、大容量メディアへの対応が必須です。たとえば、新発売の光磁気ディスク(CZ-6MO1)と光磁気ディスクカートリッジ(JY-701MPA)なら、ディスク1枚で65,536色画像にして1,000枚強、15.6kHzの音声サンプリングデータで約20時間強もの情報を記憶できます。絵に書いた餅とされていた「画像データベース」も、「サンプリングシステム」さえも、もう実用レベル。SCSIの採用が、夢の大容量メディアに応えてくれるからです。

X68000の先見性

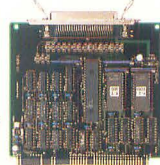
初代X68000は、すでにハードディスクインターフェイスを内蔵していたこと。当時また一般的ではなかったハードディスクに対して先見の発想で臨んでいたわけです。今回のSCSI対応も同様、100MBを超える大容量メディアハンドリングがスタンダードになる日も、そう遠くはありません。

大容量ハードディスクか? 光磁気ディスクか? それとも……

考えもしなかった新しいデバイスか。新製品X68000SUPERのSCSIインターフェイスに何を接続するかは、賢明なユーザー諸兄にお任せするとして…。このマシンがまた新たな一歩を踏み出したことに異論はないはずです。蛇足ながらこのSUPERシリーズに関していわせてもらえれば、その日から大容量ハンドリングをお望みの方にはSUPER HDを。未来に夢を託したユーザーはSUPER、といったところでしょうか。

＊SCSI装置をご使用の場合は、Human68k Ver2.0以上でご使用ください。
＊ビジュアルシェル上からはSCSI装置はご使用になれません。

●デイジーチェーン方式による接続例

SCSIボードCZ-6BS1▶
標準価格29,800円(税別)

敢えてX68000の大容量メディア対応を実証する意味。

カラー画像ファイル、サンプリングファイルへ。
X68000のクリエイティブユースに待望の大容量ファイル。
書き換え可能な光磁気ディスク、登場。

■光磁気ディスクユニット■
CZ-6MO1…標準価格450,000円(税別)

■光磁気ディスクカートリッジ■
JY-701MPA…標準価格30,000円(税別)

写真のX68000とディスプレイは別売です。

SX-WINDOW、SCSIインターフェイス標準装備。

SUPER
本体+キーボード+マウス・トラックボール
CZ-604C-TN(チタンブラック) 標準価格348,000円(税別) NEW
HDタイプ CZ-623C-TN(チタンブラック) 標準価格498,000円(税別)

EXPERT II
本体+キーボード+マウス・トラックボール
CZ-603C-BK(ブラック)・-GY(グレー) 標準価格338,000円(税別)
HDタイプ CZ-613C-BK(ブラック) 標準価格448,000円(税別)

PRO II
本体+キーボード+マウス
CZ-653C-BK(ブラック)・-GY(グレー) 標準価格285,000円(税別)
HDタイプ CZ-663C-BK(ブラック)・-GY(グレー) 標準価格395,000円(税別)

DISPLAY LINE UP

- 15型カラーディスプレイテレビ(ドットピッチ0.39mm) CZ-602D-BK(ブラック)・-GY(グレー)※ 標準価格99,800円(チルトスタンド同梱・税別)
- 15型カラーディスプレイテレビ(ドットピッチ0.39mm) CZ-605D-BK(ブラック)・-GY(グレー) 標準価格115,000円(スピーカー2個/チルトスタンド同梱・税別)
- 15型カラーディスプレイテレビ(ドットピッチ0.31mm) CZ-613D-TN(チタンブラック)・-BK(ブラック)・-GY(グレー) 標準価格135,000円(スピーカー2個/チルトスタンド同梱・税別)
- 14型カラーディスプレイ(ドットピッチ0.31mm) CZ-603D-BK(ブラック)・-GY(グレー) 標準価格84,800円(チルトスタンド同梱・税別)
- 14型カラーディスプレイ(ドットピッチ0.31mm) CZ-604D-BK(ブラック)・-GY(グレー) 標準価格94,800円(スピーカー2個/チルトスタンド同梱・税別)
- 14型カラーディスプレイ(ドットピッチ0.31mm) CZ-606D-TN(チタンブラック)・-BK(ブラック)・-GY(グレー) 標準価格79,800円(チルトスタンド同梱・税別) NEW
- 21型カラーディスプレイ(ドットピッチ0.52mm) CU-21HD-BK(ブラック) 標準価格148,000円(スピーカー2個同梱・税別)

※印の商品は在庫僅少です。

X68000
自分流カード
デザインコンペ
作品大募集

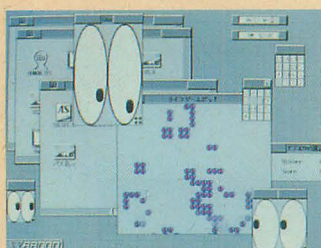
〈応募要領〉●応募方法/X68000で作成したポストカードサイズのデザインカードを送って下さい。(ソフトは自由)●作品分類/部門A:クリスマスカード、ニューイヤーカード 部門B:バレンタインカード、バースディカード 部門C:暑中見舞カード、サークル・趣味の会お知らせカード●賞/A・B・C各部門毎に優秀作品を選考、オリジナルカレンダーに掲載してプレゼントします。※優秀作品賞:掲載作品応募者に、カレンダー及びオリジナル表彰楯を進呈。※参加賞:応募者全員に、カレンダーを進呈。(応募作品に関わる諸権利は主催者に帰属するものとして作品の返却はいたしません)
●応募期間/1990年10月1日～1991年2月28日(消印有効)

詳細はX68000販売店店頭で、チラシ・ポスターをご覧下さい。

●お問い合わせは…
シャープ株式会社
電子機器事業本部システム機器営業部
〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部液晶映像システム事業部第2商品企画部
〒162 東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)
東京は平成3年1月より番号が変わります。☎(03)3260-1161(大代表)

Oh!X

C O N T

〈スタッフ〉
●編集長／前田 徹 ●副編集長／植木章夫 ●編集／岡崎栄子 浅井研二 ●協力／有田隆也 中森 章 後藤貴行 林 一樹 荻窪 圭 岡本浩一郎 毛内俊行 吉田賢司 影山裕昭 相馬英智 古村 聡 村田敏幸 丹 明彦 三沢和彦 長沢淳博 宮島 靖 金子俊一 浦川博之 山田純二 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 小栗由香 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 AD GREEN ●校正／グループごじら

特集 急接近! SX-WINDOW

特別付録より VS2.X

DōGA・CGアニメーション講座

ソル・フィース

銀河英雄伝説II

光磁気ディスクCZ-6MO1

表紙絵：塚田哲也

1991 JAN. 1

E N T S

■広告目次

SHARP システムパフォーマンスを実証する多彩なペリフェラル

ディスプレイ関連

カラーディスプレイテレビ

15型カラーディスプレイテレビ
CZ-602D-BK
★CZ-602D-GY
標準価格 99,800円(税別)
(チルトスタンド同梱)

15型カラーディスプレイテレビ
CZ-605D-BK・-GY
標準価格115,000円(税別)
(スピーカー2個・チルトスタンド同梱)

15型カラーディスプレイテレビ
CZ-613D-TN・-BK・-GY
標準価格135,000円(税別)
(スピーカー2個・チルトスタンド同梱)

CRTフィルター

高性能CRTフィルター
BF-68PRO
標準価格 19,800円(税別)
(14/15型用)

カラーディスプレイ

14型カラーディスプレイ
CZ-606D-TN・-BK・-GY
標準価格79,800円(税別)
(チルトスタンド同梱)

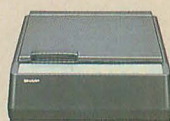
14型カラーディスプレイ
CZ-604D-BK・-GY
標準価格 94,800円(税別)
(スピーカー2個・チルトスタンド同梱)

21型カラーディスプレイ
CU-21HD
標準価格 148,000円(税別)
(スピーカー2個同梱)

チューナー

RGBシステムチューナー
CZ-6TU-BK・-GY
標準価格 33,100円(税別)
(リモコン付)

アートツール

画像入力

カラーイメージスキャナ※1
CZ-8NS1
標準価格 188,000円(税別)

スキャナ用パラレルボード
CZ-6BN1
標準価格 29,800円(税別)

映像入力

カラーイメージユニット※2
CZ-6VT1-BK
CZ-6VT1
標準価格 69,800円(税別)

映像出力

ビデオボード※3
CZ-6BV1
標準価格 21,000円(税別)

プリンタ

熱転写カラープリンタ

48ドット
熱転写カラー漢字プリンタ
★CZ-8PC4
CZ-8PC4-GY
標準価格 99,800円(税別)
(信号ケーブル同梱)

カラービデオプリンタ

カラービデオプリンタ
★ CZ-6PV1
標準価格 198,000円(税別)
(信号ケーブル同梱)

カラーイメージジェット

カラーイメージジェット※4
IO-735X
標準価格248,000円(税別)
(信号ケーブル別売)

カラードットプリンタ

24ピン
カラー漢字プリンタ(80桁)
CZ-8PG1
標準価格 130,000円(税別)
(信号ケーブル同梱)

24ピン
カラー漢字プリンタ(136桁)
CZ-8PG2
標準価格 160,000円(税別)
(信号ケーブル同梱)

ドットプリンタ

24ピン漢字プリンタ(136桁)
CZ-8PK10
標準価格 97,800円(税別)
(信号ケーブル同梱)

ファイル

光磁気ディスク

光磁気ディスクユニット※5
(594MB)
CZ-6MO1
標準価格450,000円(税別)
(SCSIケーブル同梱)
※光磁気ディスクカートリッ
は別売です。別売のJY-7
MPA 標準価格30,000円
(税別)をご使用ください。

ハードディスク

ハードディスクユニット(20MB)
CZ-620H
標準価格 178,000円(税別

増設用ハードディスク
ドライブ (40MB)
(CZ-602C/603C/652C/
653C内蔵用)
CZ-64H
標準価格 120,000円(税別
(取付費別
※取付に関してはシャープ
お客様ご相談窓口にてご
相談ください。

※1 ご使用に際しては、カラーイメージスキャナCZ-8NS1に同梱のRS-232Cケーブルで接続するか、より高速のパラレルデータ伝送を行う場合、別売のスキャナ用パラレルボードCZ-6BN1標準価格29,800円(税別)で接続してください。
※2 CZ-603D/604D、CU-21HDをご使用の場合は、RGBシステムチューナーCZ-6TU(別売)が必要です。 ※3 ビデオ出力は15.75kHzテレビ標準信号です。また、拡張I/Oスロットは2スロット使用します。
※4 別売の信号ケーブルIO-73CX標準価格5,500円(税別)で接続して下さい。 ※5 CZ-600C、601C、602C、603C、611C、612C、613C、652C、653C、662C、663Cに使用の場合は、別売のSCSIボード(CZ-6BS1)が必要です。(但し、CZ-623Cは不要
また、X68000用OS Human68K ver.2.0以上にてご使用ください。(光磁気ディスクカートリッジは別売のJY-701MPA標準価格30,000円(税別)をご使用ください。) ※6 ご使用に際しては、あらかじめ別売の1MB増設RAMボードCZ-6BE1 標準価格

X1・X1turbo シリーズ用 周辺機器

標準価格は税別です。

カラーディスプレイ		
●21型カラーディスプレイ※1	CU-21HD	148,000円

映像・画像入力編集装置		
●カラーイメージスキャナ	CZ-8NS1	188,000円
●カラーイメージボードII	CZ-8BV2	39,800円

●立体映像セット	★CZ-8BR1	29,800円
●パーソナルテロッパ※2	CZ-8DT2	44,800円

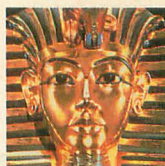
FM音源		
●ステレオタイプFM音源ボード	CZ-8BS1	23,800円

スピーカー(2本1組)標準装備、ミュージックツール同梱

プリンタ		
●24ピンカラー漢字プリンタ(80桁)	CZ-8PG1	130,000円
●24ピンカラー漢字プリンタ(136桁)	CZ-8PG2	160,000円
●24ピン漢字プリンタ(136桁)	CZ-8PK10	97,800円
●48ドット熱転写カラー漢字プリンタ	★CZ-8PC4	99,800円
●48ドット熱転写カラー漢字プリンタ	CZ-8PC4-GY	99,800円
●カラービデオプリンタ	★CZ-6PV1	198,000円
●カラーイメージジェット	IO-735X	248,000円

ファイル		
●ミニフロッピーディスクユニット(2HD・2D)※3	★CZ-520F	118,000円

シャープ株式会社 ●お問い合わせは…シャープ(株)電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大

お望みのパワーシステムへ。

シャープペリフェラルファミリー X68000

ボード

拡張メモリ

1MB増設RAMボード
（CZ-600C専用）
CZ-6BE1
標準価格 35,000円（税別）

1MB増設RAMボード
（CZ-601C/611C/652C/653C/662C/663C用）
CZ-6BE1B
標準価格 28,000円（税別）

2MB増設RAMボード ※6
CZ-6BE2
標準価格 79,800円（税別）

4MB増設RAMボード ※6
CZ-6BE4
標準価格 138,000円（税別）

インターフェイス

ユニバーサルI/Oボード
CZ-6BU1
標準価格 39,800円（税別）

GP-IBボード
CZ-6BG1
標準価格 59,800円（税別）

増設用RS-232Cボード
（2チャンネル）
CZ-6BF1
標準価格 49,800円（税別）

NEW

SCSIボード ※7
CZ-6BS1
標準価格 29,800円（税別）
（ソフトウェア（SCSIユーティリティ）同梱）

数値演算プロセッサ

数値演算プロセッサボード
CZ-6BP1
標準価格 79,800円（税別）

FAX

FAXボード
CZ-6BC1
標準価格 79,800円（税別）

MIDI

MIDIボード
CZ-6BM1
標準価格 26,800円（税別）

ネットワーク

モデム

モデムユニット ※8
CZ-8TM2
標準価格 49,800円（税別）
（RS-232Cケーブル同梱）

RS-232Cケーブル

RS-232Cケーブル
（平行接続型）
CZ-8LM1
標準価格 7,200円（税別）

RS-232Cケーブル
（クロス接続型）
CZ-8LM2
標準価格 7,200円（税別）

LANボード

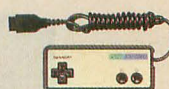
LANボード
CZ-6BL1
標準価格268,000円（税別）
（イーサネット用）
CZ-6BL2 NEW
標準価格298,000円（税別）
（イーサネット/チーパネット両用）
※電源ユニット・ソフトウェア
（ネットワークドライバVer1.0）同梱

入力

インテリジェントコントローラ
CZ-8NJ2
標準価格 23,800円（税別）

マウス・トラックボール
CZ-8NM3
標準価格9,800円（税別）

トラックボール
CZ-8NT1
標準価格 13,800円（税別）

マウス
CZ-8NM2A
標準価格 6,800円（税別）

ジョイカード
CZ-8NJ1
標準価格 1,700円（税別）

その他

拡張スロット

拡張I/Oボックス（4スロット）
（CZ-600C/601C/602C/603C/611C/612C/613C/623C用）
CZ-6EB1-BK
CZ-6EB1
標準価格 88,000円（税別）

スピーカー

アンプ内蔵
スピーカーシステム（2本1組）
AN-S100
標準価格 36,600円（税別）

システムラック

システムラック
（CZ-600C/601C/602C/603C/611C/612C/613C/623C用）
CZ-6SD1
標準価格 44,800円（税別）

35,000円（税別・CZ-600C用）、CZ-6BE1B 標準価格28,000円（税別・CZ-601C、CZ-611C、652C、653C、662C、663C用）を増設してください。 ※7 CZ-600C、601C、602C、603C、611C、612C、613Cに装着の場合、I/Oスロット2に装着ください。CZ-652C、653C、662C、663Cに装着の場合はI/Oスロット4に装着ください。また、CZ-6BG1、6BU1、6BL1、6BL2、6BN1などのボードは、接続コネクタとの関係で本ボードとの併用はできませんのでご注意ください。なお、本ボードはX68000用OS Human68K ver.2.0以上にてご使用ください。 ※8 モデムユニットCZ-8TM2に同梱のソフトはX1/X1ターボシリーズ用です。

品名	品番	価格
●ミニフロッピーディスクユニット（2D）	★CZ-502F	99,800円
●ミニフロッピーディスクユニット（2D・1ドライブ）	CZ-503F	49,800円
●増設用ミニフロッピーディスクドライブ（2D）※4	CZ-53F-BK	19,800円
拡張ボード・その他		
●モデムユニット（300/1200ボー）	CZ-8TM2	49,800円
●320KB外部メモリ	CZ-8BE2	29,800円
●RS-232C・マウスボード※5	CZ-8BM2	19,800円
●フロッピーディスクインターフェイス※6	CZ-8BF1	14,800円

品名	品番	価格
●JIS第1水準漢字ROM※7	CZ-8BK2	19,800円
●RS-232C用ケーブル（平行接続型）	CZ-8LM1	7,200円
●RS-232C用ケーブル（クロス接続型）	CZ-8LM2	7,200円
●拡張I/Oボックス	CZ-8EB3	33,800円
●RFコンバータ※8	AN-58C	2,980円
●インテリジェントコントローラ	CZ-8NJ2	23,800円
●マウス・トラックボール	CZ-8NM3	9,800円
●マウス	CZ-8NM2A	6,800円
●トラックボール	CZ-8NT1	13,800円

品名	品番	価格
●ジョイカード	CZ-8NJ1	1,700円
●チルトスタンド	CZ-6ST1-E・-B	5,800円
●高性能CRTフィルター※9	BF-68PRO	19,800円
●スキャナ用パラレルボード※10	CZ-8BN1	27,800円

●品番中の-表示は、B〈ブラック〉・E〈オフィスグレー〉を示します。※1 X1ターボZシリーズ用 ※2 CZ-862Cには接続できません。※3 X1ターボシリーズ用 ※4 CZ-830C用 ※5 X1シリーズ用 ※6 CZ-850CでCZ-520Fを使用する場合に必要 ※7 CZ-800C、801C、802C、803C、811C、820C用 ※8 CZ-820C、822C、830C用 ※9 14/15型用 ※10 CZ-8NS1用●接続等の説明につきましては、周辺機器総合カタログをご参照ください。

★印の商品は在庫僅少です。

機器事業本部液晶映像システム事業部第2商品企画部 〒162 東京都新宿区市谷八幡町8番地☎(03)260-1161（大代表） 平成3年1月より以下におかけください。☎(03)3260-1161（大代表）

ハイアビリティを実証する多彩なソフトウェア。

ドロー編集、WYSIWYG印刷、こんなC.G.ツールが欲しかった。

本格的なロゴタイプやPOPを簡単に作成できるグラフィックツールです。優先順位が任意に指定できるドローセル、ペイントセル、テキストセルの3つの仮想セルで、目的にあった自由なグラフィックが駆使できます。また印刷は、画面イメージがそのまま印刷イメージとなるWYSIWYG(What You See Is What You Get)を実現。A6/A5/A4/A3/B6/B5/B4/葉書サイズで8色カラー印字できます。

〈ドローセル〉ベジェ曲線によって少ないデータ量でも複雑な絵を描くことができます。エンベロープ変形を始めとした豊富な編集機能を持っており、拡大、縮小しても絵の美しさは変わりません。またテキストセルで作成したベクトルフォントデータを自由に変形し、オリジナルロゴタイプやPOPを作成できます。

〈ペイントセル〉ペンやエアーブラシ、ペンキなどを使って、ピクセルで構成されたビットマップ図形を描くことができます。また、「NEW PrintShop PRO-68K」や「X-BASIC」、「Z's STAFF PRO-68K」のデータ取り込みやイメージスキャナによる取り込みをサポートしています。

〈テキストセル〉通常の文字入力機能に加え、ベースライン変形などの多彩な編集機能によって自由に文字の加工ができます。また英数文字のベクトルフォントを標準装備。さらに「Z's STAFF PRO-68K Ver2.0」、「書体倶楽部」の日本語ベクトルフォントが利用可能。また、内蔵の漢字ROMフォントも自動的にベクトルフォントデータに変換しますので、簡単に日本語ロゴタイプを作成することができます。

※「Z's STAFF PRO-68K」、「書体倶楽部」は、㈱Zeitの製品です。
※本ソフトの動作には、メインメモリ2MBが必要です。

CANVAS PRO-68K
CZ-249GS 標準価格29,800円(税別)

●主として個人用のさまざまなジャンルのデータが収められているドローグラフィックデータ集です。

海のデータ/動物のデータ/スポーツのデータ/鳥のデータ/人物のデータ/食物のデータ/昆虫のデータ

CANVAS PRO-68K ドローグラフィックライブラリ VOL.1
CZ-255GS 標準価格8,800円(税別)

●主としてビジネス用のさまざまなジャンルのデータが収められているドローグラフィックデータ集です。

OA関係のデータ/飾りのデータ/コンピュータ関係のデータ/POPのデータ/国旗のデータ/字体のデータ/地図のデータ/乗り物のデータ

CANVAS PRO-68K ドローグラフィックライブラリ VOL.2
CZ-256GS 標準価格8,800円(税別)

バージョンアップされたCコンパイラと、強力なBASTOCチェッカー。

ソースコードデバッガをはじめ、各種開発ツールを強化。バージョンアップされたCコンパイラ。

Cのソースレベルでデバッグできる「ソースコードデバッガ」を搭載したほか、各種開発ツールを強化した総合開発ツールです。また、ライブラリはHuman 68k ver2.0の拡張DOSコールもサポートしているなど、よりX68000のハードウェアを活かせる豊富なライブラリ(830種以上)となっています。C言語の標準であるANSI規格準拠をさらに強化。「プログラム保守ユーティリティ(MAKE)」や「ライブラリアン」など各種ツールを追加しました。その他「BASIC-Cコンバータ」、「アセンブラ」、「リンカ」、「デバッガ」、「ソースコードデバッガ」、「アーカイバ」、「コンバータ」、などのツールが装備されています。

※C compiler PRO-68K(CZ-211LS)を既にお持ちの方は、登録カードをもとに有償バージョンアップを行います。
※本ソフトの動作にはメインメモリ2MBが必要です。

CZ-245LS 標準価格44,800円(税別)

トラブルエラーの悩み解消!
「XBAStoC」の強力ツールの登場です。

X-BASICプログラムのコンパイル時、発見しづらいトラブルエラーに悩まされていたプログラムの問題点をひとつひとつ指摘。エラーとなる直接原因だけでなく、注意項目も指摘します。これにより、X-BASICでは実行できたのにコンパイルするとエラーが発生する、といったプログラムの修正が簡単にできます。

●指摘したトラブルの結果を、画面やプリンタなどの外部デバイスに簡単に出力できます。●エラーラインとエラーレポート、2つのエラーファイルを自動的に生成。●グラフィカルな画面による簡単操作。●コマンドラインからダイレクトに操作を指定。バッチファイルに組み込むなどの修正作業の自動化が可能。●GP-IBボード(CZ-6BG1)とユニバーサルI/Oボード(CZ-6BU1)付属の拡張外部関数もコンパイル可能。

※X-BASICプログラムをコンパイルするためには、別売の「C compiler PRO-68K」(CZ-211LS)または「C compiler PRO-68K ver2.0」(CZ-245LS)が必要です。

XBAStoC CHECKER PRO-68K
CZ-260LS 標準価格9,800円(税別)

シャープ株式会社 ●お問い合わせは……シャープ㈱電子機器事業本部液晶映像システム事業部第2商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161(大代表)へ。

資料請求券 CZソフト Oh!X 1体

シャープオリジナルソフトウェア
X68000

お望みのワークベンチへ。

ビジネスツール

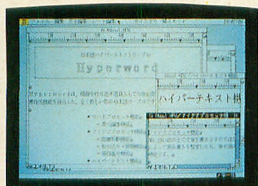
Hyperword
■CZ-251BS 標準価格39,800円(税別)

X68000の優れたグラフィック環境を活用し効率的に文書を作成するためのインテリジェントワープロです。アイデアプロセッサ機能、ハイパーテキスト機能などをサポート。データの整理やプレゼンテーションツールなど幅広い用途に利用できます。

CYBERNOTE PRO-68K
■CZ-243BS 標準価格19,800円(税別)

プライベートなデータやビジネスデータを簡単な操作で管理・運営できるパーソナルデータベースです。リフィル、タックシール、ハガキなどへの印字もOK。シャープ電子手帳とのデータ交換可能(別売の通信ケーブルCE-300Lが必要)。

Stationery PRO-68K
■CZ-240BS 標準価格14,800円(税別)

他のソフトを起動する前に、このStationery PRO-68Kを一度起動するだけで、他のソフトを実行中にも「スケジュール」「住所録」など多彩な機能をワンタッチで使用できます。シャープ電子手帳とのデータ送受信も実現。(別売の通信ケーブルCE-300Lが必要)。

TOP給与計算エキスパート
■CZ-228BS 標準価格200,000円(税別)

給与計算から明細発行までを、リアルイメージ入力により自動的に、素早く処理することができます。

TOP財務会計
■CZ-227BS 標準価格200,000円(税別)

会計エキスパートシステムとデータベースを搭載し、機能と操作性を両立させた財務会計ソフト。

CARD PRO-68K
■CZ-226BS 標準価格29,800円(税別)

自由なレイアウト画面で入力できるワープロ機能を装備したカード型リレーショナルデータベース。

CARD PRO-68K用システム手帳リフィル集
■CZ-241BS 標準価格9,800円(税別)

CARD PRO-68K用活用フォーム集
■CZ-242BS 標準価格9,800円(税別)

DATA PRO-68K
■CZ-220BS 標準価格58,000円(税別)

入力の手間を軽減するヒストリー機能を装備した、コマンド型リレーショナルデータベースです。

BUSINESS PRO-68K
■CZ-212BS 標準価格68,000円(税別)

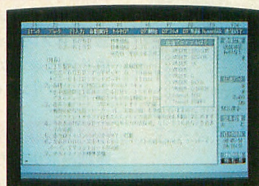
スプレッドシート(表計算)、データベース、グラフ作成機能を一体化させた統合ビジネスツールです。

アートツール

NEW PrintShop PRO-68K
■CZ-221HS 標準価格19,800円(税別)

オリジナリティあふれるはがき等、簡単に作成、印刷できるホームプロダクティビリティツール。

グラフィックライブラリ VOL.1
■CZ-235GS 標準価格8,800円(税別)

グラフィックライブラリ VOL.2
■CZ-236GS 標準価格8,800円(税別)

通信ツール

Communication PRO-68K ver2.0
■CZ-257CS 標準価格19,800円(税別)

Communication PRO-68Kのバージョンアップ版です。MNPモデムへの対応で、ハードフロー制御(CTS/RTS)をサポートしています。

※バージョンアップ対応中。

開発ツール

SX-WINDOW ver1.0
■CZ-259SS 標準価格6,800円(税別)

複数の作業を同時に処理できる疑似マルチタスクや入出力装置の設定が簡単に行える多機能コントロールパネルを搭載した本格ウィンドウシステムです。IOCSコールを利用したソフトの処理速度を高速化するIOCS.Xを付属。

OS-9/X68000
■CZ-219SS 標準価格29,800円(税別)

マルチタスク機能、リアルタイム機能を活かした使いやすく機能的なOS環境を提供します。

※OS-9はマイクロウェア社の登録商標です。

Human68k ver2.0
■CZ-244SS 標準価格9,800円(税別)

システムパフォーマンスをさらに高める処理機能を付加したHuman68kの最新バージョンです。

THE福袋V2.0
■CZ-224LS 標準価格9,980円(税別)

AI-68K(Staff LISP/OPS PRO-68K)
■CZ-234LS 標準価格188,000円(税別)

サウンドツール

Musicstudio PRO-68K ver.1.1
■CZ-252MS 標準価格28,800円(税別)

MUSIC PRO-68K〔MIDI〕
■CZ-247MS 標準価格28,800円(税別)

ソングライブラリ〈101曲集〉
■CZ-248MS 標準価格8,800円(税別)

Sampling PRO-68K
■CZ-215MS 標準価格17,800円(税別)

SOUND PRO-68K
■CZ-214MS 標準価格15,800円(税別)

MUSIC PRO-68K
■CZ-213MS 標準価格18,800円(税別)

シューティングゲーム
〈ツインビー〉
■CZ-217AS
標準価格7,800円(税別)
©KONAMI 1988

シューティングゲーム
〈沙羅曼蛇〉
■CZ-218AS
標準価格8,800円(税別)
©KONAMI 1989

ブロックゲーム
〈アルカノイド〉
■CZ-222AS
標準価格7,800円(税別)
©TAITO CORP. 1987

ドライブゲーム
〈フルスロットル〉
■CZ-231AS
標準価格8,800円(税別)
©TAITO CORP. 1988

スポーツゲーム
〈熱血高校ドッジボール部〉
■CZ-232AS
標準価格7,800円(税別)
©TECHNOS JAPAN CORP. 1988

アクションゲーム
〈パックマニア〉
■CZ-233AS
標準価格7,800円(税別)
©NAMCO

アクションゲーム
〈ニュージーランドストーリー〉
■CZ-230AS
標準価格8,800円(税別)
©TAITO CORP. 1989

スポーツゲーム
〈V'BALL〉
■CZ-246AS
標準価格7,900円(税別)
©TECHNOS JAPAN CORP. 1989

バイクレーシングゲーム
〈スーパーハングオン〉
■CZ-238AS
標準価格8,800円(税別)
©SEGA 1987

ジェットヘリ・シミュレーションゲーム
〈サンダーブレード〉
■CZ-239AS
標準価格9,500円(税別)
©SEGA 1987

アクションゲーム
〈ダウンタウン熱血物語〉
■CZ-254AS
標準価格8,800円(税別)
©TECHNOS JAPAN CORP. 1989

アクションゲーム
〈サイバリオン〉
■CZ-229AS
標準価格8,800円(税別)
©TAITO CORP. 1988

スポーツゲーム
〈熱血高校ドッジボール部サッカー編〉
■CZ-262AS
標準価格8,800円(税別)
©TECHNOS JAPAN CORP. 1990

平成3年1月より以下におかけください。☎(03)3260-1161(大代表)

HAL Laboratory, inc.

X68000対応

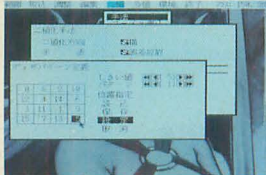
X68専用、初のハンディスキャナ

▶驚異の256階調

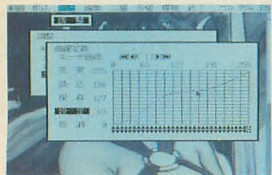
Fine Scanner-X68

HGS-68 39,800円(税別)

取り込み用イラストサンプル付

【取り込み】取り込み中は画面がスクロールしてリアルタイムに表示されます。表示は64階調グレイスケールですが内部データは256階調になっています。

【二値化】取り込み画像を二値化してペインティングの境界をはっきりさせます。二値化手法には誤差拡散をはじめ各種ディザやパターンを自由に定義できるユーザディザも用意しました。

【調整】明るさ・コントラスト・γ補正などの各種調整は、取り込み終了後に思のまま変化させることができます。また、調整しながら結果が画面で瞬時に確認できます。

【組み込み】ここでは、「Z's STAFF PRO-68K」に組み込んでみました。付属ソフトでは多彩なファイル形式をサポートしていますので、ほとんどのグラフィックツールで使用可能です。

【γ補正】ディスプレイやプリンタなどの微妙な輝度特性を補正するためのγ補正。予め用意された補正カーブに加えてユーザが自由にγカーブを定義できる「ユーザ曲線定義」を用意。

【彩色】ペインティングを施した取り込み画面はこんなに美しく仕上がりました。ファインスキャナを使えば簡単な操作でグラフィックの世界を満喫できます。

※画面は、開発中のもので製品とは異なる場合があります。

対応機種

■シャープ X68000シリーズ
(1Mモデルの場合、機能の一部に制約があります。)

動作環境

■使用OS:Human68k
■ディスプレイ:シャープ専用モニター(あるいは同等品)
■その他●マウス、マウストラックボール、トラックボール対応●プリンタ(CZ/PC-PR/NM/VP系)対応

スキャナ

■読み取り方式:ハンディタイプ■読み取り幅:105mm■移動速度:20mm/秒■使用光源:黄緑色LED■階調:256階調グレイスケール■解像度:100/200dpi■インターフェース:専用インターフェースボード(拡張I/Oスロットを使用)■外形寸法:幅127×奥行140×高さ30mm■重量:250g

付属ソフト

■Image Photo 68k●範囲●取り込み:範囲内、自由長、取込位置表示●調整:明るさ、コントラスト、γ補正(ディスプレイ、プリンタ、HALFAX、ユーザ曲線定義)●編集:複写、回転、クリア、修正●二値:二値化手法(誤差拡散、ディザ、ユーザディザパタン)、拡大、縮小、印刷(原寸印刷可能)、画面表示、ファイル作成●多値(グレイスケール):画面表示(64階調/16階調グレイスケール)、ファイル作成・ファイル読み込み●環境設定:画面モード選択、画面色設定、スキャナ選択、ポート番号設定、プリンタ選択、環境保存

対応ファイル形式

■二値●べた形式、拡張べた形式、ZIM(圧縮/非圧縮)形式、GL0形式、GL3形式、Prizm(非圧縮)形式■多値(グレイスケール)●Image Photo形式、拡張Image Photo形式、ZIM(圧縮/非圧縮)形式、GL0形式、GL3形式、Prizm(非圧縮)形式PIC形式(書込みのみ)

商品構成

●スキャナ本体●制御ボード●ACアダプタ●フロッピーディスク(サポートソフト)●取扱説明書●保証書●ご愛用者カード

ご注意

●印刷は誤差拡散などの手法で二値化して行ないます。グレイスケール画像の印刷を行なうにはビデオプリンタなどが必要です。
●各アプリケーションソフトに組み込んだ場合、ファイル構造による制約のため256階調で表現できないものがあります。

グレイスケールとは

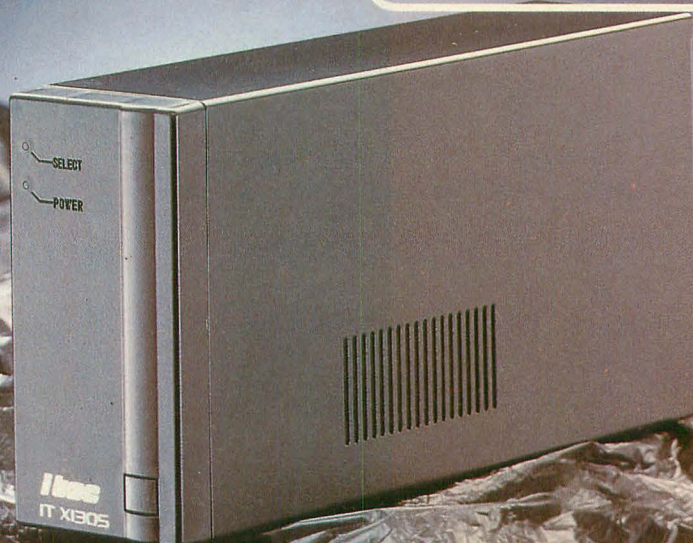
今までのスキャナの主流は中間調を白か黒の点の密集度によって明るさを表現していましたが、ファインスキャナではドット単位に256階調の明るさ情報をもっています。このような方法をグレイスケールといいます。

※Human68kはシャープおよびHudsonの商標です。
※商品名は、各社の商標または登録商標です。
※商品構成、仕様につきましては予告なく変更することがあります。

対応ソフト

●Z's STAFF PRO-68K ㈱ツァイト●マジックパレット ㈲ミュージカル・プラン●Prism ㈱ウルフ・チーム●G 68K VersionII-PRO ㈱SYTEM HOUSE OH!

株式会社HAL研究所
〒101 東京都千代田区神田須田町2-6-5 OS'85ビル5F ☎03-252-5561㈹

'91 NEW WAVE
itec
HARD DISK UNIT
SHARP X68000専用
テックのハードディスクがさらに進化しました。
ARP X68000専用SCSIハードディスクユニットIT X80S
X130Sの登場です。
テックでは早くからX68000専用ハードディスクに着手し今ま
SASIタイプハードディスクに多大なる評価を得てきました。
技術をさらに進化させた結果次代が求めるSCSIタイプハー
ィスクの発表となったわけです。
000 SUPER HDの発売によりSCSIのもつ拡張性能の良
あらためて認識されるようになりました。また従来のX68000
ーズにもSHARP製SCSIインターフェースボードCZ-6BS1
いることにより完全にSCSI機器を周辺装置として接続が可
なりました。アイテックのSCSIハードディスクはこのどち
対応できるもので最大7台デージチェーンでの接続、また
大容量80MB/130MBさらに60×120×243(㎜)のコンパクト
S IT X130Sをあなたはどのように
SELECT
POWER
itec
IT X130S
SELECT
POWER
itec
IT X80S
IT X80S ¥128,000
国産高精度SCSIドライブ搭載
容量80MB
平均アクセスタイム20ms
接続ケーブル添付
IT X130S ¥158,000
国産高精度SCSIドライブ搭載
大容量130MB
平均アクセスタイム20ms
接続ケーブル添付
アイテック株式会社
本社 〒550 大阪市西区新町1丁目25番14号 TEL:(06)532-0216 FAX:(06)532-7253
関西営業所 〒550 大阪市西区新町1丁目25番14号 TEL:(06)532-0120 FAX:(06)532-0543
関東営業所 〒275 千葉県習志野市谷津1丁目12番5号 TEL:(0474)77-7564 FAX:(0474)73-2759
名古屋営業所 〒460 名古屋市中区大須2丁目28番31号 TEL:(052)212-1487 FAX:(052)212-1627
インフォメーションセンター
TEL:(06)532-0320

レイトレーシングソフト

C-TRACE+(プラス)、新発売。

メタボール対応

メタボール…
有機的な質感を表現する事ができます。他のプリミティブと論理演算ができます。

αチャンネル機能…
αチャンネルへの対応により、高度な合成作業が可能となります。

スコープ機能…
一度生成した画像の部分修正が可能となります。アニメーションへの応用により、作業の大幅な効率化が図れます。

スポット光源への対応…
照射範囲の設定が自由にできます。さらに照射範囲の境界のぼかしも可能です。

DOS-EXTENDERへの対応…
CPUがV30のユーザーの方も使用できるようにノーマル版もバンドリング。(98版のみ)

対応機種…
NEC PC9801シリーズ
EPSON PC286シリーズ
EPSON PC386シリーズ
シャープ X68000シリーズ

定価…¥198,000

※ワイヤフレームで物体を確認できるワイヤ・ビュー搭載予定。

レイトレーシングを高速にしたい方へ。

C-TRACE TP Ver.3.0 ¥298,000

トランスピュータボード(T-800×1+4M)+
C-TRACE Ver.3.0トランスピュータ版ソフトウェア 〈ソフト+ハード〉
68000+C-TRACE Ver.3.0のスピードの〝約170倍〟
80386+C-TRACE Ver.3.0のスピードの〝約40倍〟
●対応機種/98版…PC9801シリーズ、または互換機なら新旧問わずほとんどの機種に対応。ただし、標準拡張スロットがない機種には、装着できません。68版…X68000全機種
★もっとスピードを上げたい方へ…並列処理によってスピードアップ可能!
増設トランスピュータモジュール(1TRAM)¥298,000

★フルカラーフレームバッファ	¥69,800
C-TRACE98 EXTENDER	¥128,000
C-TRACE98 Ver.3.0	¥98,000
C-TRACE68 Ver.3.0	¥98,000
C-TRACE TOWNS	¥68,000
C-TRACE NEWS Ver.3.0	¥530,000
★C-TRACE98 TP Ver.3.0	¥298,000
★C-TRACE68 TP Ver.3.0	¥298,000

標示価格に消費税は含みません。★の製品は店頭販売いたしておりません。直接当社まで、お申し込みください。
クレジット可

バージョンアップ受付中。

●お問い合せ先　株式会社キャスト
〒158 東京都世田谷区等々力2-1-13
TEL.03-705-1065 FAX.03-705-5224

第2回サイクロンCG大会記念ビデオ発売!!

応募作品紹介　実務利用法　テクニカル解説　商品情報

価格¥3,000(税別)

サイクロンExpressα［エクスプレスアルファ］好評発売中!!

CG画像制作センター
秋葉原サテライト開設
(03)839-8481

●主な業務内容/CG画像制作プロデュース・アプリケーション開発受託
サイクロンユーザー会ネットワークサポート・3次元CAD×CGシステム
導入コンサルタント及び教育・アウトプットサービス等々

藍津 享さん「時」第2回サイクロンCG大会グランプリ受賞

松塚 哲さん「AKANE'90「希」」ソフトバンク賞受賞　山本 健介さん「分校 放課後」特別努力賞受賞　丸口 正史さん「NIGHT(ナイト)」NEC賞受賞　稲見 薫さん「午後(暮れない日)」優秀賞受賞　田村 順司さん「水杯によせて」富士通賞受賞

3D・CG レイトレーシング サイクロンがパソコンCGの世界を越えた

●サイクロンExpressαでは、旧バージョンで実現したボクセル分割の技術をポリゴン対応へと拡張しました。ボクセル分割は物体の数があるラインに達すると、それ以後はほとんど処理時間が増加しないという、抜本的な高速化技術です。これにより、数千、数万のポリゴンであっても十分実用的な時間内で処理することが可能になりました。また、マッピング機能の充実で表現力がさらにアップ！テクスチュアマッピングはもとより、凹凸表現を行うバンプマッピングや、1つの物体に貼りつけることによって複数の質感表現ができるアトリビュートマッピングが使用可能です。またそれぞれについて、1個の物体に対し最大5枚の多重マッピングが施せる等、自由度が大変高くなっています。

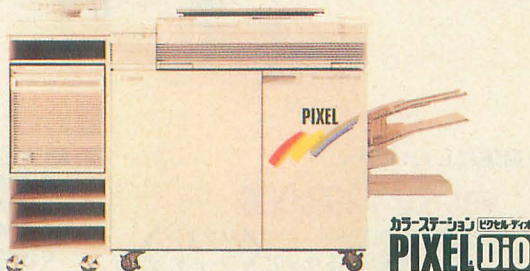
●ネイティブモード　サポートアプリケーション
(NEC PC-9801RA/RLバージョンのみに対応版別途30,000円)
PC-9801上で80386CPUをネイティブモードで直接ドライブすることによって、メモリー空間の使用可能範囲は、最大14メガバイトまでの拡大が可能。大変スケールの大きな物の処理が可能になりました。

●TOWNS版—「Z's TRIPHONY DIGITAL CRAFT」とのデータコンバータ装備版　12月発売予定

サイクロンExpressα98…………165,000円より
(NEC PC-9801VX、RX、RA)
※フレームバッファ(スーパーフレームorハイパーフレーム)が必要です。

サイクロンExpressα68…………98,000円
(SHARP X68000)

サイクロンExpressαTOWNS…98,000円
(富士通 FM-TOWNS)

サイクロンExpressαはキャノンの高画質カラーステーション(ピクセル・ディオ)に対応!!

※別途GPIBボード、connect:GPIB(ライフボート社)が必要となります。

彩ネット(BBS)開局!! 会員募集中
300〜9600bPS MNP5オンラインサインアップあり。詳しくはお問い合せください。
092-526-0158[彩ネット1]　03-839-0448[彩ネット2]

九州本社 〒810 福岡市中央区平丘町68　phone.092-522-6347　FAX.092-521-0400
秋葉原サテライト 〒101 東京都千代田区6-3-8　phone.03-839-8481　FAX.03-839-8438

勇者たちよ！いま一度力を…
Dungeon Master
CHAOS STRIKES BACK
12月21日発売
新たな挑戦の幕開けが迫る！
世界をそして日本を興奮の渦に巻き込んだリアルタイムRPG「ダンジョンマスター」その続篇が帰ってきた！「ロード・カオス」を倒し世界に平和と秩序をもたらした勇者達…。しかし「ロード・カオス」は生きていた。あの時、彼は既に自分が勇者達に倒されることを知り、秘かに新しいダンジョンを作って悪の力を蓄えていたのだ。
アニメを使ったイントロダクション、FM音源対応による音楽とサウンド、チャンピオンの顔を自由に書き換えることのできるキャラクターエディット機能、さらには日本版独自のプレイヤーの現在地を一目で表示するマップ表示機能や新しい魔法の追加など熱中度さらにパワーアップ。
前作で育てた勇者をそのまま使って冒険に旅立つもよし、新たに用意された勇者を編成してダンジョンに向かうもよし、ドキドキしながらこの興奮を味わって下さい。前作以上の難しさとおもしろさだけは保証します。
続 ダンジョン・マスター カオスの逆襲
●X68000版 ¥9,800(税抜)
c 1990, SOFTWARE HEAVEN, INC./FTL GAMES.
LICENSED THROUGH AN AFFILIATION WITH J.P. INTERNATIONAL.
c 1990 VICTOR MUSICAL INDUSTRIES, INC.
NEURAL GEAR
シューティングの極み!!
SCORE 34500
SCORE 140200
HIT COUNT 68
SCORE 282700
HIT COUNT 27
驚異の迫力で展開する変化に富んだ全10ステージ。
各ステージ毎に武器(5種類)の選択、ステージの解説の表示をし、その間のプログラムロードにより途中のディスクアクセスがなく、スムーズにプレイできます。
大型エネルギー・ゲージ、スコアー・ゲージの採用による迫力ある戦闘がたのしめます。
戦闘中のスピード変更機能搭載による状況に応じた臨機応変の戦いを実現。
快感のテーマ曲ほか全20曲収録。内蔵音源に加えてMIDI音源にも対応。
迫力のサンプリング効果音も搭載。
ゲーム性、グラフィックス、サウンド何もかもがX68000の限界を超えた!!
好評発売中!
本格的3D快感シューティングゲーム
ニューラル・ギア
●X68000対応 ¥8,800(税抜)
企画・開発:Fill in Cafe
発売:ビクター音楽産業株式会社 〒151 東京都渋谷区千駄ヶ谷2-8-16 (03)423-7901代

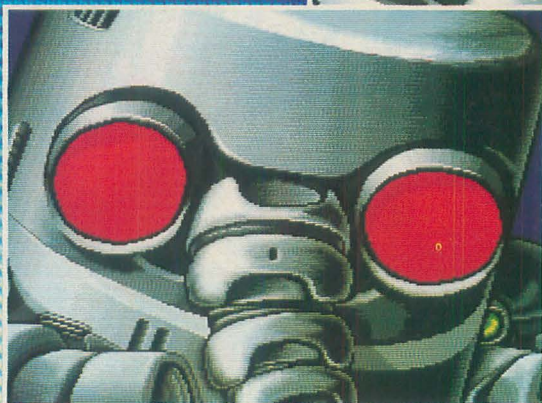
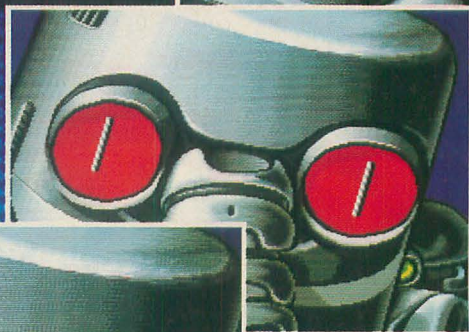
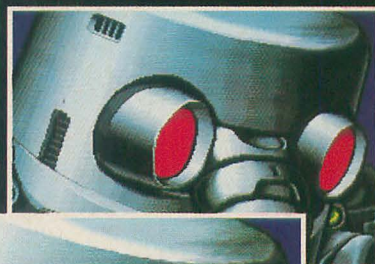
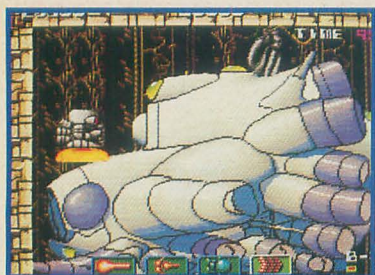
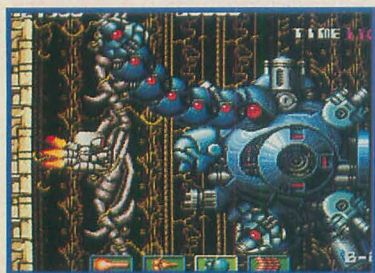
ATOMIC ROBO-KID

アトミック・ロボキッド

全方向スクロール

ロボットシューティングの極限!

アーケード版アトミックロボキッド、
X68000に登場!!
'90年年末、ついにその全貌を現わす。

君はこのボスキャラたちにどう立ち向かうか!?

2XXX年、核戦争の難を逃れた少数の人々は、地下に隠れて放射能の嵐がやむのを待ち、地上に出た。しかし、人々のDNA(染色体)は、わずかに残った放射能によって破壊され、人類は自分達の子孫を残せなくなってしまう。トミタ博士は、わずかに残された人々をシェルターに冷凍保存しDNA正常化のプログラムを開発する。ところが、シェルターに向かうロボキッドが動き出す直前に博士は、死んだ。自分の目的も解らずに、目覚めたロボキッド。……はたして、ロボキッドは人類を救う事ができるのか?

新発売!!

X68000
5"-2HD

●ローランド社　MT-32、CM32L 完全対応
MIDIインターフェイスボードC-Z-6BMI
又は、SACOM製SX-68Mが必要です。
標準価格8,800円　Copyright ©UPL

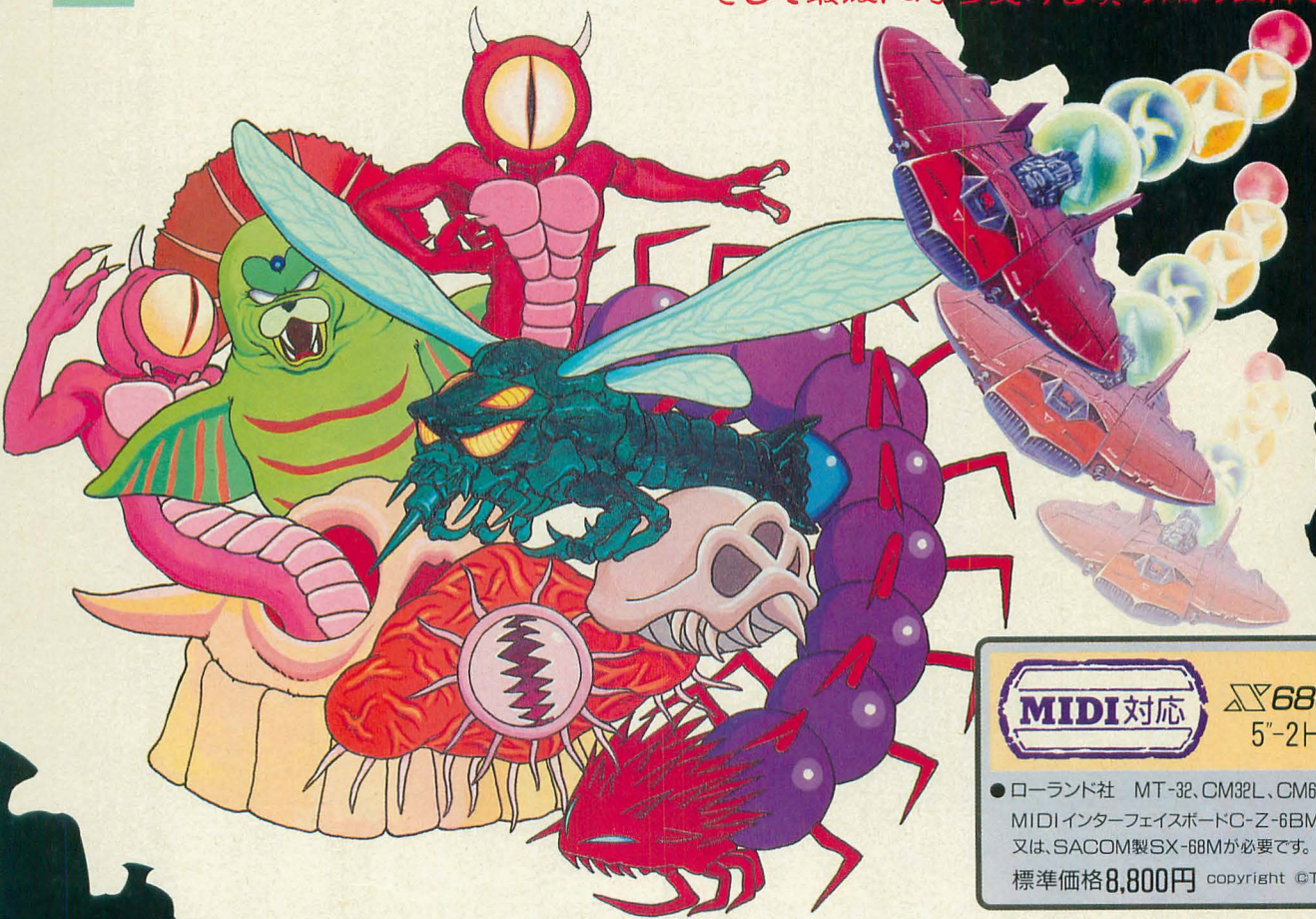
ログインNo.23のP170〜P173に
「徹底解剖!!」が掲載されているので注目!!

Gemini Wing™

ジェミニウィング

アドレナリン全開シューティング
ジェミニウィング
大ヒット驀(ばく)進中!!

ガンシップ(スクランブル)緊急発進!!

君は迫り来る「蟲」の恐怖に耐えられるか?
そして最後に待ち受ける真の敵の正体とは?

MIDI対応

X68000
5"-2HD

● ローランド社　MT-32、CM32L、CM64完全対応
MIDIインターフェイスボードC-Z-6BMI
又は、SACOM製SX-68Mが必要です。

標準価格8,800円　copyright ©TECMO

※標準価格に消費税は含まれておりません。

Information

「サコムタイムズ」が発行されます。全国のパソコンショップにて、ただ今、配布中!!
お近くに、パソコンショップがない場合、下記の住所に72円切手同封の上お申し込み下さい。
〒130 東京都墨田区両国4-38-16
両国桜井ビル4F ㈱システムサコム

ROUND2

髑髏山の雷神
ツインギドル

ROUND4

絶海の怒り
バルファ

ROUND5

幻の幽霊戦艦
スカルフェルド

株式会社 システムサコム
〒130 東京都墨田区両国4-38-16
両国桜井ビル4F
ソフトウェア部03(635)7609

※通信販売(送料サービス):品名・機種・住所・氏名・電話番号を明記して、
現金書留でお申し込みください。

ACTION GAME

DYNAMITE DUKE

ダイナマイト・デューク

奴の野望は、俺が打ち砕く‼

西暦2089年、オゾン層の破壊による環境悪化により、人類は滅亡の危機に瀕していた。
この事態を憂慮した各国の指導者は、世界連合軍を設立、厳しい環境下にも耐えうる人体の研究開発に乗りだす。
そして、ついにバイオテクノロジーの力によって、強化人間を実現させたのである。
しかし、開発に関わった一人の大佐が技術を盗み出し、強化人間をも超える存在を創り上げることに成功。
……、彼の目的は、軍隊を組織し人類を支配することであった！
連合軍は、この野望を打ち砕くため、ひとりの強化人間を送り出した。
機械の右腕を持つ男、陸軍特殊部隊大佐デューク・フリードリッヒ・フェルゼン、コード・ネーム＝Red・Dynamite。
しかし、人は彼のことを、こう呼んだ「ダイナマイト・デューク‼」。

●大人気アーケード・ゲームからの移植！●シューティングの興奮＋アクションの一体感！●アメリカン・コミック調の美しい画面！●アニメのように派手に動ド迫力デカ・キャラ！●変化に富んだ各ミッションと個性あふれる敵キャラ！●必殺の一撃「ダイナマイト・パンチ」の快感！

好評発売中‼ ——— X68000専用(ジョイスティック、キーボード対応FM音源対応1PLAYER専用) 5"2HD 3枚組 ¥8,800(税別

株式会社ヘルツ
〒169 東京都新宿区北新宿2-1-16松本ビル3号2F TEL.03(371)3012

STAFF募集‼
企画、プログラマー、デザイナー、サウンドクリエイター
詳細は電話連絡にて。担当秋山まで。

X68000シリーズに、新登場！
シュヴァルツシルト
12月7日発売。

ストーリィ性を持ったドラマティックなゲーム展開

シュヴァルツシルトの最大の特徴は、そのゲームシステムにあります。単なるウォーシミュレーションではなく、ゲームを進めていくにしたがって、次々に新たな目的が現われ、プレイヤーは知らず知らずにゲームのシナリオに引き込まれていくという、ドラマティックなゲーム展開が魅力の、SFシミュレーションゲームです。

究極のゲームシナリオ

ゲームのおもしろさはシナリオで決まります。軍事行動、外交政策調査・研究、資金運用、商業取り引きといった戦略要素を完璧にシミュレート。シミュレーションゲームの面白さを徹底的に追求した究極のゲームシナリオです。

●5"2HD・3枚組¥12,800(価格には消費税は含まれておりません

〒162 東京都新宿区市谷台町1
TEL. 03-353-7724

全国通販

SHARP 認定 PPO-SHOP **O.A.ランド**

(TEL) 03-770-8855

■アフターサービス万全のサポート体制
- 下取・買取は電話で見積りしております。責任を持って下取りさせて頂きます。
- ご注文、お問合せは…。午前10時から午後7時まで
- 商品のお届けは…入金確認後、即日発送致します。

■TEL・FAXのお見積OK.!!
■低金利クレジットをご利用下さい。

▶12・15〜1・14

SHARPのことならなんでもおまかせ!!

ボーナス・シーズン大徳買セール! 安く値切ってネ。
お電話下さい。㊙価格をお知らせいたします。

流通事情により、広告表示価格は、お安くなる場合がありますので、ドンドンお電話下さい。

CYBER STICK
■CZ-8NJ2
(定価￥23,800)
OAランド特価 ▶￥18,000

MIDIインターフェースボード
サコム(定価￥19,800)
■SX-68M
OAランド特価 ▶￥15,800
●X68000専用純正コンパチ

SHARP X68000シリーズセット どんどんTEL下さい。

X68000 SUPER NEW

①CZ-604C-TN+CZ-613D-TN
定価合計￥483,000

1回	355,000	12回	32,100
24回	16,900	36回	11,700

②CZ-604C-TN+CZ-606D-TN
定価合計￥427,800

1回	315,000	12回	28,400
24回	15,000	36回	10,400

■CZ-604C
特価~~￥348,000~~
■CZ-623C
特価~~￥498,000~~

X68000 SUPER-HD

①CZ-623C-TN+CZ-613D-TN
定価合計￥633,000

1回	466,000	12回	42,100
24回	22,200	36回	15,400

②CZ-623C-TN+CZ-606D-TN
定価合計￥577,800

1回	425,000	12回	38,400
24回	20,300	36回	14,000

X68000EXPERT-II

①CZ-603C+CZ-613D
定価合計￥473,000

1回	348,000	12回	31,400
24回	16,600	36回	11,500

②CZ-603C+CZ-605D
定価合計￥453,000

1回	333,000	12回	30,100
24回	15,900	36回	11,000

③CZ-603C+CZ-606D
定価合計￥417,800

1回	TEL下さい。	12回	27,800
24回	14,600	36回	10,100

■CZ-603C
特価~~￥249,000~~
■CZ-613C
特価~~￥448,000~~

X68000 EXPERT II-HD

①CZ-613C+CZ-613D
定価合計￥583,000

1回	TEL下さい。	12回	38,800
24回	20,500	36回	14,200

②CZ-613C+CZ-605D
定価合計￥563,000

1回	414,000	12回	37,400
24回	19,500	36回	13,700

③CZ-613C+CZ-606D
定価合計￥527,800

1回	TEL下さい。	12回	35,100
24回	18,500	36回	12,800

X68000 PROII

①CZ-653C+CZ-613D
定価合計￥420,000

1回	288,000	12回	26,000
24回	13,700	36回	9,500

②CZ-653C+CZ-605D
定価合計￥400,000

1回	273,000	12回	24,700
24回	13,000	36回	9,000

③CZ-653C+CZ-606D
定価合計￥364,800

1回	TEL下さい。	12回	22,300
24回	11,800	36回	8,200

■CZ-653C
特価~~￥285,000~~
■CZ-663C
特価~~￥395,000~~

X68000 PROII-HD

①CZ-663C+CZ-613D
定価合計￥530,000

1回	TEL下さい。	12回	35,200
24回	18,600	36回	12,900

②CZ-663C+CZ-605D
定価合計￥510,000

1回	TEL下さい。	12回	33,900
24回	17,900	36回	12,400

③CZ-663C+CZ-606D
定価合計￥474,800

1回	TEL下さい。	12回	31,600
24回	16,600	36回	11,500

■期間中、セットでお買い上げの方には、① Vボール ② ニュージーランド・ストーリー(ゲーム)のがついてきます。さらに、③ テトリスやドルアーガの塔などの入ったゲームパックもプレゼント.!!

周辺機器コーナー 電話で値切ろう。

プリンターセットコーナー

①CZ-8PC4(GY)(48ドット/カラー対応/ハガキ可能)
定価￥99,800……特価~~￥64,800~~

②CZ-8PK10(24ピン漢字プリンター136桁)
定価￥97,800……特価~~￥78,000~~

③CZ-8PG1(24ピンカラー漢字プリンター80桁)
定価￥130,000…特価~~￥103,000~~

④CZ-8PG2(24ピンカラー漢字プリンター136桁)
定価￥160,000…特価~~￥125,000~~

OAランド特選品.!!

■IO-735X(定価￥248,000)
●カラーイメージジェットプリンター
特価~~￥190,000~~

モデム

オムロン	MD-1200A III	￥14,500
	MD-24FP4 II	￥27,500
	MD-24FP5 II	￥29,800
	MD-24FN4	￥28,000
	MD-24FN5	￥31,300
	MD-24FJ4	￥31,300
	MD-24FJ5	￥34,500
	MD-24FS4	￥28,500
	MD-24FS5	￥34,500
アイワ	PV-A24VM5	￥32,500
	PV-M24	￥28,800
NEC	COMSTAR 2424/4	￥28,800
	COMSTAR 2424/5	￥33,500

X68000用周辺機器コーナー

①CZ-6VT1(カラーイメージユニット)
定価￥69,800……特価￥52,500
②CZ-8NS1(カラーイメージスキャナー)
定価￥88,000……特価￥141,000
③CZ-6BM1(MIDIボード)
定価￥26,800……特価￥20,500
④CZ-8NJ2(インテリジェント・コントローラー)
定価￥23,800……特価￥18,000
⑤CZ-6TU(RGBシステムチューナー)
定価￥33,100……特価￥25,000
⑥CZ-64H(増設ハードディスク)
定価￥120,000……特価￥90,000
⑦CZ-6EB1(拡張I/Oボックス=4スロット)
定価￥88,000……特価￥66,000
⑧CZ-6BP1(数値演算プロセッサボード)
定価￥79,800……特価￥60,000

■I・O DATA 増設RAMボード

●1MB増設RAMボード
PIO-6BE1-A
定価￥25,000

特価￥19,000

●2MB増設RAMボード
PIO-6BE2-2M
定価￥50,000

特価￥36,000

●4MB増設RAMボード
PIO-6BE4-4M
定価￥88,000

特価￥63,000

OAランド今月の大目玉!!=超A級中古品

☆限定3セットのみ.!!
■EXPERT-II
- CZ-603C 新品
- CZ-605D 新品
- [CZ-8NS1 中古 / CZ-6BN1 中古 (パラレルボード)]
- PIO-6BE2 新品
- SX-68M(MIDI) 新品
- CZ-8NJ2 新品

OAランド特価 ￥455,000

今がチャンス!!

☆限定2セットのみ.!!
■SUPER-HD
- CZ-623C-TN 新品
- CZ-613D-TN 新品
- PIO-6BE4 新品
- CZ-8PG2 中古
- CZ-245LS 中古 (Cコンパイラ II)

OAランド特価 ￥557,000

上記セットでなくとも、希望の単体商品だけでも販売致しますので、TEL下さい。早目に☎をチョーダイ。すぐなくなりますです。ハイ。

新製品 周辺機器

20%くらいは、引いちゃうヨ.!!

■光磁気ディスクユニット
●CZ-6MO1
(定価￥450,000)
特価~~￥450,000~~

■SCSIボード
●CZ-6BS1
(定価￥29,800)
特価~~￥29,800~~

■XBAS to C CHECKER PRO68K
●CZ-260LS(定価￥9,800)
特価~~￥9,800~~

OAランド推奨 周辺機器

■SX-WINDOW
(次代インテリジェント・ソフト)
●多機能コントロールパネル搭載の本格ウィンドウシステム。
定価￥6,800
特価￥5,100

■CZ-6BV-1
(ビデオボード)
●ビデオ出力は、テレビ標準信号、拡張I/Oスロット使用
定価￥21,000
特価￥15,800

■CZ-245LS
(C-コンパイラ)
●ソースコードデバッガをはじめ、各種開発ツールを強化。II版
定価￥44,800
特価￥34,000

■ゲームソフト、ビジネスソフト2割、3割引きは、当り前で〜す。お問い合せ下さい。

通信販売のご案内

—— 全国通販 ——

■銀行振込で申し込みの方は商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。

〔振込先〕第一勧業銀行 渋谷支店
普通No.1163457 ㈱オーエーランド

品川、目黒方面 / 首都高速3号線 / O.A.ランド / 道玄坂 / 渋谷駅 / 井の頭線渋谷駅 / ヤマハ / J&P / 109 / 神泉駅 / 明治通り / 山手線 / 東急百貨店 / 西武百貨店

■12月より年内無休.!!

■現金書留で送金されるお客様は電話番号と商品名、数量を明記して同封して下さい。■クレジットでご購入を希望される方は申し込み用紙をお送り致しますのでご記入の上返送して下さい。20才以上の方は、原則として保証人不要です。クレジットは1〜60回払で月々5,000円よりご自由に設定できます。

クレジット表

3回	3%	6回	4%	10回	5.5%	12回	5.5%	15回	8%	18回	10%	20回	11%
24回	11.5%	30回	15.5%	36回	16%	42回	20.5%	48回	21%	54回	26.5%	60回	27%

㈱オー・エー・ランド

〒150 東京都渋谷区円山町20-4 第5日新ビル1F
☎(03)770-8855 FAX(03)770-7080

関東エリアの送料は、1個につき￥1,000です。

★全商品保証書付。専門のアドバイザーが、お客様のニーズに対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。

■表示価格は、税別表示です。詳しくは、お電話にて、お問い合せ下さい。掲載の価格は、11月下現在です。
(注)■平成3年1月より、電話番号が変わります。▶03-3770-8855です。宜しくお願い致します。

ちゃんとパソコンしたい人の。
TSUKUMO

もう、離れられなくなるね。

ツクモイメージガール
越智静香

あなたに、会いたい。

12月15日(土) OPEN AM10:15

うわさのパソコンロフト「ツクモパソコン本店」オープン

日本で初めての総合カメラ専門店「ツクモAV/カメラ館」はオープンセール中

PRESENT ツクモ全店で1万円以上お買い上げの方先着1万名様に越智静香チャンのフロッピーカレンダーをさしあげます。

静香チャンに会おう! 12月24日(月)PM1:00〜パソコン本店3Fのイベントフロアにて越智静香チャンのサイン会が開かれます。

ツクモパソコン本店
〒101 東京都千代田区外神田1-9-7 ☎03-253-5599

ツクモAV/カメラ館
〒101 東京都千代田区外神田1-11-3 ☎03-254-3999

23(金)〜1/6(日)の期間中に¥5,000以上お買い上げのお客様に、恒例**秋葉原電気まつり**のチケットを差し上げます。(今年は賞金総額¥7,000万円)

A Happy New Open!

ツクモAV/カメラ館とツクモパソコン本店はX68000の宝島!どうぞおこし下さい。

掲載商品2万円以上送料無料!!

ツクモの3つのNEW　1 新しい年!　2 新しい店!　3 新しい商品!

シャープX68000コーナーが更に充実!! ツクモパソコン本店にX68000コーナーが移転し、4倍の広さに拡大!! 今まで以上にソフトからハードに致るまで更に充実、是非お立ち寄り下さい。

各種・各社新製品も大特価販売中!詳しくはお問い合せ下さい。

X68000 シリーズ

新・製・品
X68000 SUPER
CZ-604C
SCSIインターフェイス内蔵タイプ
★オプションの80MB内蔵ハードディスクを追加する事によってSUPER HDに変身!

アイテックのSCSIハードディスクとのお買得なセットがございますのでお尋ね下さい。

	型番	定価
PRO II	CZ-653C	定価¥285,000
	CZ-663C	定価¥395,000
EXPERT II	CZ-603C	定価¥338,000
	CZ-613C	定価¥448,000
SUPER HD	CZ-623C	定価¥498,000

本店オープン記念特価にて提供中!! 是非、お尋ねください。

年末年始 各種プリンタ大奉仕!
超お買得
一流メーカー品　台数限定
★48ドット熱転写カラープリンタ
Happy特価¥59,800(消費税別途¥1,794)
★ドットマトリクス漢字プリンタ(80桁)
Happy特価¥39,800(消費税別途¥1,194)
★ドットマトリクス漢字プリンタ(136桁)
Happy特価¥59,800(消費税別途¥1,794)

パワーアップアイテム!更に安く強力に!ハードディスク

標準タイプハードディスク(SASI)
アイテック IT X640(40MB)
Happy特価¥84,800
(消費税別途¥2,544)
アイテック IT X680(80MB)
Happy特価¥99,800
(消費税別途¥2,994)

(カラー:ブラック、グレー)

SCSIハードディスク
アイテック IT X80S(80MB)
定価¥128,000　Happy特価¥99,800
アイテック IT X130S(130MB)
定価¥158,000　Happy特価¥125,800
※X68000SUPER以外の機種は、CZ-6BS1(SCSIボード 定価¥29,800)が必要です

ハードディスク目玉品
40MB SASIタイプ
Happy特価
¥59,800
(消費税別途¥1,794)

光磁気ディスクユニット　台数限定
ソニー　NWP-539N(光磁気ディスクユニット)¥440,000
シャープ　CZ-6BS1(SCSIボード)……¥29,800
SCSIケーブル……¥10,000
光磁気ディスク……サービス(¥30,000)
合計定価¥509,800
Happy特価¥408,000(消費税別途¥12,240)
クレジット例(48回払・税込)初回¥13,110+月々¥11,300×47回

ミュージックツール
セット
CM-32L 定価¥69,000
SX-68M 定価¥19,800
Musicstudio Mu-1 定価¥19,800
合計定価¥108,600
Happy特価 ¥88,000
(消費税別途¥2,640)
クレジット例(18回払・税込)
初回¥7,223+月々¥5,600×17回

セット
CM-64 定価¥129,000
SX-68M 定価¥19,800
Musicstudio Mu-1 定価¥19,800
合計定価¥168,600
Happy特価 ¥138,000
(消費税別途¥4,140)
クレジット例(18回払・税込)
初回¥9,318+月々¥8,900×17回

アートツール
[ハードウェア]
一流メーカーA4サイズイメージスキャナ
Happy特価¥128,000 台数限定 (消費税別途¥3,840)
ビデオボード CZ-6BV1……定価¥21,000
カラーイメージユニット CZ-6VT1……定価¥69,800
数値演算プロセッサボード CZ-6BP1……定価¥79,800
[ソフトウェア]
CANVAS PRO-68K 定価¥29,800 ツクモ特価販売中!!
Z's STAFF PRO-68K Ver.2.0
Happy特価¥49,300 (消費税別途¥1,479)
マジックパレット
Happy特価¥16,800 (消費税別途¥504)
彩クロンExpress α68
Happy特価¥83,300 (消費税別途¥2,499)
デジタルクラフト
Happy特価¥38,800 (消費税別途¥1,164)

情報ツール
電子手帳シリーズ
ハイパー電子システム手帳
PA-9500 新製品
定価¥48,000
●表計算カード PA-9C1……定価¥16,000
●RAMカード PA-9C90(64KB)……定価¥14,000
●RAMカード PA-9C91(128KB)……定価¥20,000
PA-8600……定価¥28,000 Happy特価¥24,800 (消費税別途¥744)
CE-300L (通信ケーブル)定価¥2,800 Happy特価¥2,500 (消費税別途¥75)
電子手帳対応ソフト
●CYBER NOTE PRO-68K……定価¥19,800
●Stationery PRO-68K……定価¥14,800

ツクモグローバルカード
大/好/評/入/会/者/募/集/!
国内・外で大活躍!
使って安心、持ってて便利!ツクモグローバルカードはジャックス・VISA、セントラル・マスターとの提携カードです。ツクモ各店でのお買物がらくらくできるうえに、国内はもとより海外でのショッピングもOK!
(03)251-9898又は各店で!
18才以上なら

メモリーボード
1MB増設RAMボード(ACE/PROシリーズ用)
Happy特価¥19,000(消費税別途¥570)
2MB増設RAMボード
Happy特価¥37,000(消費税別途¥1,110)
4MB増設RAMボード
Happy特価¥64,000(消費税別途¥1,920)
※計測技研のメモリーボードも取り扱っております。価格については、おたずねください。

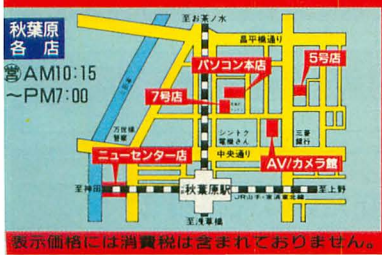
コミュニケーションツール
通信ソフト
●た〜みのる2 定価¥17,800……Happy特価¥15,000
●Communication PRO-68K Ver.2.0 定価¥19,800
モデム
●一流メーカー2400ボークラス5……Happy特価¥29,000

ビジネスツール
●Hyper WORD……定価¥39,800 Happy特価
●CARD PRO-68K……定価¥29,800 販売中!!
●Kamikaze……Happy特価¥57,800

開発ツール
●C Compiler PRO-68K Ver.2.0……定価¥44,800
●SX-WINDOW……定価¥6,800
●XBAS to C CHECKER PRO-68K……定価¥9,800

ポケットコンピューター
PC-E550
Happy特価¥28,800
(消費税別途¥864)
通信ケーブルCE-140T
Happy特価¥8,800
(消費税別途¥264)

ツクモ通販センター フリーダイヤル受注専用 0120-377-999
商品についてのお問い合せは各店店頭又は、☎03(251)9911へ。

秋葉原各店
営 AM10:15〜PM7:00
至お茶ノ水　昌平橋通り　パソコン本店　5号店　7号店　万世橋警察　シントク電機さん　三菱銀行　ニューセンター店　中央通り　AV/カメラ館　至神田　秋葉原駅　JR山手・京浜東北線　至上野　至浅草橋
表示価格には消費税は含まれておりません。

ツクモは「スーパーX PRO SHOP」です。
PRO STAFF ツクモ
九十九電機株
〒101-91 東京都千代田区神田郵便局私書箱135号
パソコン本店 荒井　N・C店 福地
★商品のご注文は在庫確認の上お願いします。

ツクモパソコン本店☎03-253-5599(担当/荒井)
便利で安心な通信販売
ツクモ通販センター☎03-251-9911
■ツクモAV/カメラ館 ☎03-254-3999(担当/川名)
■ツクモニューセンター店 ☎03-251-0987(担当/福地)
■ツクモ5号店 ☎03-251-0531(担当/森)
■名古屋1号店 ☎052-263-1655(担当/吉高)
■名古屋2号店 ☎052-251-3399(担当/佐原)
■ツクモ札幌 ☎011-241-2299(担当/田口)

カード払い	全国代金引き換え配達	クレジット払い	現金書留払い	銀行振込払い	各種リース払い
通信販売での御利用カード、ツクモグローバルカード、VIPカード、セントラル、ジャックス※御本人様より電話で通信販売部へお申し込み下さい。	お申し込みは☎03-251-9911へお電話1本!配達日の指定もできます。	月々¥3,000以上の均等払いも頭金なし、夏・冬ボーナス2回払いも受付中!	〒101-91 東京都千代田区神田郵便局私書箱135号 ツクモ通販センター Oh!X係	事前に☎でお届け先をご連絡下さい。富士銀行 神田支店(普)No.894047 ツクモデンキ	くわしくは各店にお問い合せ下さい。ケースに合わせてご相談にのらせて頂きます。

東京各店電話番号変更のおしらせ　平成3年1月1日より東京各店の電話番号の局番の頭に3がつきます。

■店頭にて、ゲームソフト25%OFF!!(税別)、超低金利 ハッピークレジットをご利用ください!!
■特に人気のある商品によっては、しばらくお待ち願うことがありますのでご了承下さい。

厳選された製品を、より安く、より早く、皆様のお手元に!!

広告掲載商品以外の製品も取扱っております。

■年末店頭大恩謝セール実施中!!ゲームソフト(ビジネス)新製品続々入荷中!!

ラストチャンス!! X68000SUPER-HDセット 今がお買徳!! 送料無料です。

●ザ・ワークステーションと呼ぶにふさわしい
SUPER-HD=オクト厳定セット!!

※マウス・トラックボール付!!
ディスプレイにはスピーカ2個
チルト台付!!

Ⓐ:CZ-623C-TN+CZ-606D-TN…………定価合計¥577,800▶大特価

12回	¥37,800	24回	¥20,000	36回	¥13,900	48回	¥10,900	60回	¥9,100

Ⓑ:CZ-623C-TN+CZ-613D-TN…………定価合計¥633,000▶大特価

12回	¥41,600	24回	¥22,000	36回	¥15,300	48回	¥12,000	60回	¥10,100

現金特価!! お電話下さい。

上記セットお買い上げの方に、オクトからのプレゼント!!
①MD-2HD 10枚
②サンダーブレード(ゲーム¥9,500)
③ジョイカード(連射式)
④シリコンキーボードカバー(¥2,800)

※超低金利クレジットをご利用下さい。1回〜60回払い、頭金ナシ!! ボーナス1回及び2回払いOKです。

X68000ソフト大セール実施中※ゲームソフトオール25%off 送料¥500

〈グラフィック〉●Z's STAFF PRO68K Ver.2.0
(シャフト)定価¥58,000………特価¥39,800
〈データーベース〉●KAMIKAZE
定価¥68,000…………特価¥45,800
〈グラフィック〉●C-TRACE68
(キャスト)定価¥68,000………特価¥51,000
〈C言語〉●C&Professional Pack
定価¥58,000…………特価¥43,800
〈グラフィック〉●サイクロン エキスプレス
定価¥78,000…………特価¥28,000
〈グラフィック〉●デジタルクラフト
定価¥39,800…………特価¥28,000
〈ワープロ〉●ハイパーワード
定価¥39,800 CZ-251BS……特価¥29,800
〈開発ツール〉●C-コンパラPRO68KV.2
定価¥44,800 CZ-245IS……特価¥33,600
〈CGツール〉●CANVAS PRO68K
定価¥29,800 CZ-249GS……特価¥22,400

型名	商品	定価	特価
CZ-212BS	BUSINESS PRO-68K	¥68,000	¥48,000
CZ-213MS	MUSIC PRO68K	¥18,800	¥13,500
CZ-214MS	SOUND PRO-68K	¥15,800	¥11,500
CZ-215MS	Sampling PRO-68K	¥17,800	¥12,800
CZ-219SS	OS-9/X68000	¥29,800	¥21,000
CZ-220BS	DATA PRO-68K	¥58,000	¥41,000
CZ-223CS	Communication PRO-68K	¥19,800	¥14,300
CZ-224LS	THE 福袋 V2.0	¥9,900	¥7,500
CZ-226BS	CARD PRO-68K	¥29,800	¥21,300
CZ-241BS	システム手帳リフィル集	¥9,800	¥7,500
CZ-242BS	活用フォーム集	¥9,800	¥7,500
CZ-244SS	Homan 68K Ver.2.0	¥9,800	¥7,500
CZ-247MS	MUSIC PRO-68K(MIDI)	¥28,800	¥20,800
CZ-240BS	Stationery PRO-68K	¥14,800	¥11,500
CZ-243BS	CYBER NOTE PRO-68K	¥19,800	¥15,200
EW		¥38,000	¥29,800
G-68K		¥14,800	¥11,400
E-68		¥19,800	¥15,300
CZ-255GS	CANVASドローグラフィックLIB	¥8,800	¥6,600
CZ-256GS	CANVASドローグラフィックVOL2	¥8,800	¥6,600
CZ-260LS	XBAS to CHECKER PRO68K	¥9,800	¥7,500
CZ-259SS	SX-WINDOW Ver.1.0	¥6,800	¥5,000
CZ-234LS	AI-68K	¥188,000	¥139,000
CZ-252MS	MUSIC STUDIO PRO68K	¥28,800	¥21,500

モデムコーナー (送料¥1,000)

オムロン
●MD-1200AIII……特価¥14,500
●MD-12FS……特価¥15,000
●MD-24FP4II……特価¥26,000
●MD-24FN5……特価¥30,000
●MD-24FS4……特価¥31,000
●MD-24FS5……特価¥34,000
●MD-24FS7……特価¥44,000
●MD-24FC5……特価¥34,000
●MD-24FP5II……特価¥28,500
●MD-24FN4……特価¥27,000
●MD-24FJ4……特価¥31,000
●MD-24EJ5……特価¥34,000
●MD-24HS……特価¥64,000
●MD-48HS……特価¥98,000
●MD-96FS5……特価¥131,000
AIWA
●PV-A24VM5……特価¥30,000
●PV-M24VM5……特価¥30,000
●PV-A12……特価¥14,500
●PV-M24……特価¥28,500

熱転写カラー漢字プリンター (ケーブル 用紙付) 送料¥1,000

■IO-735X
●カラーイメージジェットプリンター

①CZ-8PK10(24ピン漢字プリンター136桁)
定価¥97,800……大特価!! TEL下さい。
②CZ-8PG1(24ピンカラー漢字プリンター80桁)
定価¥130,000……大特価!! TEL下さい。
③CZ-8PG2(24ピンカラー漢字プリンター136桁)
定価¥160,000……大特価!! TEL下さい。
④IO-735×(カラーイメージジェット)
定価¥248,000……大特価!! TEL下さい。

パソコンラック 推奨 送料無料

①五段キャスター付
5段キャスター付
キーボードが収納できるから、手元でマウス操作がラクラクできる
棚板5段のマルチに活用できるディスク。
ウーン、こいつはデキル!
1325(H)×640(W)×700(D)
特価¥15,500

②四段キャスター付
4段キャスター付
どんなパソコンにもフレキシブルに対応!
使い易いデスクです。
1245(H)×614(W)×600(D)
特価¥11,500

③三段キャスター付
3段キャスター付
場所を選ばない簡易で便利なディスクです。
限定
1175(H)×640(W)×600(D)
特価¥8,800

周辺機器コーナー (送料¥1,000)

●CZ-6BE1 IBM増設RAMボード……(¥35,000)▶特価¥26,500
●CZ-6BE1B IMB増設RAMボード……(¥28,000)▶特価¥21,000
●CZ-6BE2 2MB増設RAMボード……(¥79,800)▶特価¥60,500
●CZ-6BE4 4MB増設RAMボード……(¥138,000)▶特価¥104,800
●CZ-6BF1 増設用RS-232Cボード……(¥49,800)▶特価¥38,500
●CZ-6BG1 GP-IBボード……(¥59,800)▶特価¥45,000
●CZ-6BM1 MIDIボード……(¥26,800)▶特価¥20,500
●CZ-6BN1 スキャナ用パラレルボード……(¥29,800)▶特価¥22,800
●CZ-6BP1 数値演算プロセッサボード……(¥79,800)▶特価¥60,500
●CZ-6BO1 ユニバーサルI/Oボード……(¥39,800)▶特価¥30,500
●CZ-6EB1/BK 拡張I/Oボックス……(¥88,000)▶特価¥66,800
●CZ-6VT1/BK カラーイメージ・ユニット……(¥69,800)▶特価¥53,000
●CZ-8NM2A マウス……(¥68,800)▶特価¥5,300
●CZ-8NT1 マウストラックボール……(¥98,800)▶特価¥7,500
●CZ-8NS1 カラーイメージスキャナ……(¥188,000)▶大特価
●CZ-6BC1 FAXボード……(¥79,800)▶特価¥60,500
●CZ-8TM2 モデムユニット……(¥49,800)▶特価¥38,000
●CZ-64H 増設ハードディスク……(¥120,000)▶大特価
●CZ-6TU GY/BK RGBシステムチューナー……(¥33,100)▶特価¥25,000
●BF-68PRO 高性能CRTフィルター……(¥19,800)▶特価¥15,500
●CZ-6MO1 光磁気ディスクユニット……(¥450,000)▶大特価
●CZ-6BS1 SCSIインターフェースボード……(¥29,800)▶特価¥22,400
●CZ-6BL2 LANボード……(¥298,000)▶大特価

特選周辺機器(送料¥1,000)

●SX-68M MIDインターフェースボード
(システムサコム)¥19,800……特価¥15,000
●CZ-6BV1 ビデオボード
¥21,000……特価¥15,700

■増設RAMボード=I・Oデータ
①PIO-6BE1-A(1MB)
¥25,000……特価¥18,000
②PIO-6BE2-2M(2MB)
¥50,000……特価¥36,300
③PIO-6BE4-4M(4MB)
¥88,000……特価¥64,000

店頭ゲームソフトオール25%off! ビジネスソフト25%より特価中

★通信販売お申込みのご案内★ 〒144 東京都大田区蒲田4-6-7 TEL:03-730-6271

お申込みはお電話でお願いします。お客様の〈住所〉〈氏名〉〈電話番号〉及び〈商品名〉をお知らせ下さい。● 入金確認後ただちに商品をご送付いたします。

現金一括払い
銀行振込:お近くの銀行より(電信扱い)にてお振込み下さい。
現金書留:封筒の中に住所・氏名・商品名をご記入の上当社までお送り下さい。

クレジット
専用お申込用紙をお送り致します。ので、必要事項をご記入、ご捺印の上ご返送下さい。手続きは簡単です。

オクト ラクラク クレジット表

1回	2.06%	3回	3%	6回	4%	10回	5.5%	12回	5.5%
15回	8%	18回	10%	20回	11%	24回	12%	30回	16%
36回	17%	48回	22%	60回	28%				

振込先
富士銀行 久ヶ原支店 ㊎No.1824
三菱銀行 蒲田支店 ㊎No.0278691
株式会社 億人(オクト)

※掲載の価格は変動しますので、まずは、お電話にてご確認ください。
※年末年始:12/31〜1/3まで休ませていただきます。
※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは電話でお問合せ下さい。
※銀行振込、または、現金書留でご注文の際には、あらかじめ電話でご確認の上、お申し込み下さい。

超低金利クレジットC

注目!!

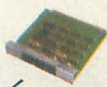
(平成3年1月末はもちろん)
平成3年2月末一括払い
手数料(金利)無料
ご利用下さい。

また また 秋葉原で おなじみ

●お近くの方は
●本体単品で
●ビジネスソフト

プリンター 10台限定 (送料¥1,000)
■CZ-8PK8 (定価¥152,000)
●24ピン漢字プリンタ(136桁)
●ハガキ印字OK!!
P&A 限定特価¥49,800
(送料・消費税込¥52,324)

CYBER STICK
●CZ-8NJ2
(定価¥23,800)
超特価!!
¥18,500 (送料・消費税込み¥19,570)

X68000シリーズ専用
MIDIインターフェースボード
SX-68M (サコム)
(純生コンパチ)定価¥19,800
特価¥16,480
送料・消費税込み!

X68000用メモリーボード (I/O・DATA) (送料¥500)

①PIO-6BE1-A 定価¥25,000……¥18,000 (送料・消費税込¥19,055)
②PIO-6BE2-2M 定価¥50,000……¥36,000 (送料・消費税込¥38,110)
③PIO-6BE4-4M 定価¥88,000……¥63,000 (送料・消費税込¥66,744)

ジョイスティック 送料¥500
●X-1PRO 定価¥9,500▶特価¥7,8
●ASCII STICK 定価¥6,800▶特価¥5,5

P&A超低金利クレジットをご利用ください!!

NEW X68000 SUPER/SUPER-HD/PROII/PROII-HD (送料・消費税込)

NEW

SUPER

セットでお買い上げの方に ●ディスケット10枚 ●ジョイカード2ケ プレゼント中!!

Ⓐセット:CZ-604C-TN+CZ-606D-TN……定価¥427,800▶特価 価格はTEL下さい。

12回	28,100	24回	14,900	36回	10,300	48回	8,100	60回	6,800

Ⓑセット:CZ-604C-TN+CZ-613D-TN……定価¥483,000▶特価 価格はTEL下さい。

12回	31,800	24回	16,800	36回	11,700	48回	9,100	60回	7,700

SUPER-HD

セットでお買い上げの方に ●ディスケット10枚 ●ジョイカード2ケ プレゼント中!!

Ⓐセット:CZ-623C-TN+CZ-606D-TN……定価¥577,800▶特価 価格はTEL下さい。

12回	37,900	24回	20,000	36回	13,900	48回	10,900	60回	9,100

Ⓑセット:CZ-623C-TN+CZ-613D-TN……定価¥633,000▶特価 価格はTEL下さい。

12回	41,600	24回	22,000	36回	15,300	48回	11,900	60回	10,000

PROII

セットでお買い上げの方に
●ディスケット10枚
●ジョイカード2ケ
プレゼント中!!

Ⓐセット:CZ-653C+CZ-606D……定価¥364,800▶特価 価格はTEL下さい。

12回	22,600	24回	11,900	36回	8,300	48回	6,500	60回	5,400

Ⓑセット:CZ-653C+CZ-605D……定価¥400,000▶特価 価格はTEL下さい。

12回	12,800	24回	13,000	36回	9,000	48回	7,100	60回	5,900

Ⓒセット:CZ-653C+CZ-604D……定価¥798,000▶特価 価格はTEL下さい。
Ⓓセット:CZ-653C+CZ-613D……定価¥420,000▶特価 価格はTEL下さい。
Ⓔセット:CZ-653C+CU-21HD……定価¥433,000▶特価 価格はTEL下さい。

PROII-HD

セットでお買い上げの方に
●ディスケット10枚
●ジョイカード2ケ
プレゼント中!!

Ⓐセット:CZ-663C+CZ-606D……定価¥474,800▶特価 価格はTEL下さい。

12回	30,800	24回	16,300	36回	11,300	48回	8,800	60回	7,400

Ⓑセット:CZ-663C+CZ-605D……定価¥510,000▶特価 価格はTEL下さい。

12回	32,500	24回	17,100	36回	11,900	48回	9,300	60回	7,800

Ⓒセット:CZ-663C+CZ-604D……定価¥489,800▶特価 価格はTEL下さい。
Ⓓセット:CZ-663C+CZ-613D……定価¥530,000▶特価 価格はTEL下さい。
Ⓔセット:CZ-663C+CU-21HD……定価¥543,000▶特価 価格はTEL下さい。

セットでお買い上げの方に
●ディスケット10枚
●ジョイカード2ケ
プレゼント中!!

◎電話にて、ドンドンお問合せ下さい!!
クレジット表には、出せないほどの価格です。
メーカーさん、ご免なさい。
ユーザーの方には大歓迎されそうです。
今がチャンスです、ハイ。

X68000シリーズ ~P&Aスペシャルセット=限定誌上販売!!

台数限定 送料、消費税込み ※セットでお買い上げの方に、●ディスケット10枚、●ジョイカード2個 プレゼント中!!

EXPERTII

Ⓐセット:P&A厳選セット
■CZ-603C
(本体価格¥338,000)
⊕
■CZ-606D
(モニター定価¥79,800)
▶P&A超特価 ¥304,000

Ⓑセット
■CZ-603C+CZ-604D
定価¥432,800…▶特価¥309,000

Ⓒセット:
■CZ-603C+DZ-605D
定価¥453,000…▶特価¥322,000

Ⓓセット:
■CZ-603C+CZ-613D
定価¥473,000…▶特価¥342,000

Ⓔセット:
■CZ-603C+CU-21HD
定価¥486,000…▶特価¥347,000

EXPERT-HD

Ⓐセット:P&A厳選セット
■CZ-612C(ブラック)
(本体価格¥466,000)
⊕
■CZ-606D(ブラック)
(モニター定価¥79,800)
▶P&A超特価 ¥335,000

Ⓑセット:
■CZ-612C+CZ-604D
定価¥560,800…▶特価¥340,000

Ⓒセット:
■CZ-612C+CZ-605D
定価¥581,000…▶超特価¥359,000

Ⓓセット:
■CZ-612C+CZ-613D
定価¥601,000…▶超特価¥372,000

Ⓔセット:
■CZ-612C+CU-21HD
定価¥614,000…▶超特価¥386,000

■モデム 限定
⊙PV-A24VM5
(アイワ)
●MNPクラス5
定価¥44,800
特価¥29,900
(送料・消費税込¥31,827)

■ALL in Note
フリートップ
パーソナルコンピュータ
⊙AX-286N-H2
(定価¥398,000)
P&A超特価
価格はTEL下さい。

●本広告の掲載の商品の価格については、消費税は含まれておりません。●営業時間=平日AM10:00~PM7:00、日祭AM10:00~PM6:00

~84回払いまでOK.!!

★頭金なし! ★即日発送

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前に、あらかじめお電話にてご確認下さい。

P&Aがズバリ超特価セールでご奉仕!!

寄り下さい。専門係員が説明いたします。
で受付します。詳しくは電話にてお問合せ下さい。
の20%引きOK! TELください。

全国通販

X68000用ソフトコーナー(送料1ヶ~5ヶまで¥500)

商品	定価	特価
Z's STAFF PRO68K Ver2.0(ツァイト)	定価¥58,000	特価¥39,500
Z's TRIPHONY デジタルクラフト(ツァイト)	定価¥39,800	特価¥27,500
テラッツォ(ハミングバード)	定価¥19,400	特価~~¥15,840~~
KAMIKAZE(サムシング・グッド)	定価¥68,800	特価¥45,500
C&Professional Pack(マイクロウェアジャパン)	定価¥58,000	特価¥43,000
Final Ver3.2(エーエスピー)	定価¥38,000	特価~~¥30,400~~
C-compiler PRO68K Ver.2 CZ-245L	定価¥44,800	特価¥34,500
CARD PRO68K CZ226BS	定価¥29,800	特価TEL下さい。
C compiler PRO68K CZ211LS	定価¥39,800	特価¥28,500
OS-9/X68000 CZ219SS	定価¥29,800	特価¥20,700
AI-68K CZ234LS	定価¥188,000	特価TEL下さい。
THE 福袋V2.0 CZ224LS	定価¥9,900	特価¥7,400
SOUND PRO68K	定価¥15,800	特価¥11,300
MUSIC PRO68K CZ213MS	定価¥15,800	特価¥13,300
Sampling PRO68K CZ215MS	定価¥17,800	特価¥12,500
MUSIC-studio PRO68K 237MS	定価¥15,800	特価TEL下さい。
MUSIC-PRO68K〔MIDI〕247MS	定価¥28,800	特価¥20,500
New-print Shop 221HS	定価¥19,800	特価TEL下さい。
Communication 223CS	定価¥19,800	特価¥9,800
Communication Ver.2 CZ-257CS	定価¥19,800	特価¥15,500
C-TRACE68 Ver.3.0(キャスト)	定価¥98,000	特価¥69,800
サイクロンEXPRESSα68	定価¥98,000	特価TEL下さい。
G68K Ver2 PRO	定価¥22,000	特価~~¥17,600~~
THE FILE PROFESSOR(ロゴシステム)	定価¥28,000	特価~~¥22,400~~
Gツール(ザインソフト)	定価¥28,000	特価~~¥22,400~~
たーみのる2(SPS)	定価¥17,800	特価~~¥14,200~~
マジックパレット(ミュージカルプラン)	定価¥19,800	特価~~¥15,800~~
Hyper word CZ-251BS	定価¥39,800	特価~~¥31,800~~

●ゲームソフト20%OFF OK.!!(一部ソフト除く)

X68000用ハードディスク(送料¥1,000)

アイテム

- ●HXD-040(40MB/23ms)……定価¥118,000▶特価¥88,000
- ●HXD-042(増設用)……定価¥128,000▶特価¥95,000

アイテック

- ●ITX-640(40MB/28ms)……定価¥158,000▶特価¥83,000
- ●ITX-680(80MB/20ms)……定価¥198,000▶特価¥97,000

プリンター(ケーブル・用紙付)限定5台 新品(送料¥1,000)

- ●CZ-8PC3(カラー漢字24ドット熱転写プリンター)
定価¥65,800……特価¥39,800
- ●CZ-8PK8(24ピン漢字プリンター136桁)
定価¥152,000……特価¥49,800
- ●CZ-8PC4 P&A特選!!(カラー漢字48ドット熱転写プリンター)
定価¥99,800……特価¥57,000

周辺機器コーナー(送料¥500)

商品	定価	特価
Ⓐ CZ-8NSI	定価¥188,000	特価¥145,000
Ⓑ CZ-6VTI	定価¥69,800	特価¥54,000
Ⓒ CZ-6TU	定価¥33,100	特価¥25,000
Ⓓ BF-68PRO	定価¥19,800	特価¥15,500
Ⓔ CZ-6BEI	定価¥35,000	特価¥26,500
Ⓕ CZ-6BEIA	定価¥38,000	特価¥28,600
Ⓖ CZ-6BE2	定価¥79,800	特価¥60,000
Ⓗ CZ-6BE4	定価¥138,000	特価¥107,000
Ⓘ CZ-6BFI	定価¥49,800	特価¥38,200
Ⓙ CZ-6BPI	定価¥79,800	特価¥61,000
Ⓚ CZ-6BMI	定価¥26,800	特価¥20,300
Ⓛ CZ-6EBI	定価¥88,000	特価¥67,500
Ⓜ AN-S100	定価¥36,600	特価¥28,500
Ⓝ CZ-6SDI	定価¥44,800	特価¥35,000
Ⓞ CZ-8PC3	定価¥65,800	P&A超特価 TEL下さい。
Ⓟ CZ-8PC4	定価¥99,800	P&A超特価 TEL下さい。
Ⓠ CZ-8PG1	定価¥130,000	P&A超特価 TEL下さい。
Ⓡ CZ-8PG2	定価¥160,000	P&A超特価 TEL下さい。
Ⓢ CZ-8PK10	定価¥97,800	P&A超特価 TEL下さい。
Ⓣ CZ-6PVI	定価¥198,000	特価¥153,000
Ⓤ IO-735X	定価¥248,000	特価¥190,000
Ⓥ CZ-8BSI	定価¥23,800	特価¥19,000
Ⓦ PIO-6BE1-A(I/O DATA)	定価¥25,000	特価¥18,000
Ⓧ PIO-6BE2-2M(I/O DATA)	定価¥50,000	特価¥36,500
Ⓨ PIO-6BE4-4M(I/O DATA)	定価¥88,000	特価¥64,300

モデムコーナー(送料¥1,000)

商品	定価	特価
Ⓐ MD-24FS5(オムロン)	定価¥49,800	特価¥34,800
Ⓑ MD-24FS7(オムロン)	定価¥64,800	特価¥45,000
Ⓒ コムスター2424/4(NEC)	定価¥38,800	特価¥28,000
Ⓓ コムスター2424/5(NEC)	定価¥44,800	特価¥32,000

P & A 特選パソコンラック (送料無料)移動自由(キャスター付)

中古パソコン(セットはモニター付)送料¥2,000

- ●X68000セット……▶¥180,000
- ●X68000 ACEセット……▶¥200,000
- ●X68000 ACE-HDセット……▶¥215,000
- ● EXPERTセット……▶¥230,000
- ● EXPERT-HDセット……▶¥265,000
- ● PROセット……▶¥250,000
- ●X68000PRO-HDセット……▶¥270,000
- ● EXPERTⅡセット……▶¥250,000
- ● EXPERTⅡ-HDセット……▶¥320,000
- ● PROⅡセット……▶¥240,000
- ● PROⅡ-HDセット……▶¥310,000

超特価でクレジットが組める!!

中古パソコンはP&Aにおまかせ!!

その場で高価現金買取り・高価下取りOK!!

■まずはお電話下さい。
03-651-1884 FAX:03-651-0141

■下取り・買取りでお急ぎの方、直接当社に来店、または、宅急便にてお送り下さい。

●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。(差額は、P&A超低金利クレジットをご利用下さい。)

●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。

●近郊の方は、P&A本店まで、直接お持ち下さい。即金にて、¥1,000,000までお支払い致します。

《便利な超低金利クレジットをご利用下さい》

- ●月々¥1,000円からOK.!!
- ●ボーナス払いOK(夏冬10回までOK)
- ●支払い回数 1回~84回
- ●お支払いは、8ヶ月先からでもOK.!!

アフターサービス万全
全商品保証付。専門の担当者がお客様の立場で対応します。
初期不良、輸送トラブルetc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

●定休日/毎週水曜日=第3水曜・木曜は連休とさせていただきます(祭日の場合は翌日になります)

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。(プリンター・フロッピーの場合、本体使用機種名を明記のこと)

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
(電信扱いでお振込み下さい。)

〔振込先〕住友銀行 新小岩支店
当No.263914 (株)ピー・アンド・エー

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回~84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

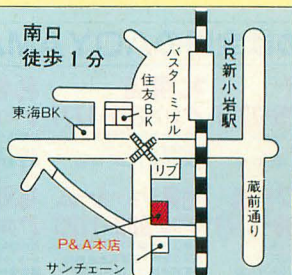
超低金利クレジット率

回数	3	6	10	12	18	24	36	48	60	72	84
手数料	3.0	4.0	5.5	5.5	10.0	11.5	16.0	21.0	27.0	35.0	42.0

●マイコン
●ビデオ
●ビデオテープ

P&A

株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-651-0148(代) FAX.03-651-0141

営業時間
平日:AM10:00~PM7:00
日祭:AM10:00~PM6:00

●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

新装オープン 11月20日 福岡ユーテック店・12月4日 池袋ソフトショップ、札幌店・12月8日 札幌PAX

LAST FESTA '90

ゼッタイ見のがせない！

X68000お買上げのお客様へ
コースで御希望ソフトは「サンダーブレード」「ダウンタウン熱血物語」「ニュージーランドストーリー」「沙羅曼蛇」「ツインビー」「フルスロットル」「ハックマニア」「ビーチバレー」「アルカノイド」「熱血高校ドッチボール」のうちいずれかからお選び下さい

買ったお客様でしかわからないこのサービス！――ぜったいX68000を買うならアオヤマがオトク
●以前当社にてX68000及びX-1を御購入いただいたお客様に限り、CZ-8PC5〔定94,800〕を大特価にてお届けいたします。会員の方は会員ダイアルにてCall！
●X68000をセットでお買い上げいただいたお客様に限り、アスキーターボステックを特価¥4,300、XE-1PROを特価¥6,700また CTRACEを特価¥39,800にて CZ-8NJ2〔インテリジェントコントローラ〕〔¥23,800〕を特価¥16,900にてお届けいたします 御注文の際に合わせてお申し込み下さい

チタンカラーの新製品登場
AOYAMA特価で更にパワーアップ

X68000PROⅡ X68000 新Bコース
CZ-653C（本体）…¥285,000
CZ-604D（0.31ステレオスピーカ付ディスプレー）…¥94,800
CZ-8NJ2（インテリジェントコントローラー）…¥23,800
ナイトアームズ…¥9,700
サンダーブレード…¥9,500
スーパーハングオン…¥7,800
パックマニア…¥7,800
ジェノサイド…¥8,800
ヴァリスⅡ…¥9,800
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト…サービス
合計¥475,000➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000PROⅡ X68000 新Eコース
CZ-653C（本体）…¥285,000
CZ-605D（0.39ステレオスピーカ付テレビ）…¥115,000
上海Ⅱ…¥6,800
倉庫番パーフェクト…¥6,800
TETRIS…¥6,800
信長の野望戦国群雄伝…¥9,800
スーパー大戦略68K…¥8,800
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト…サービス
合計¥457,000➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000EXPERTⅡ X68000 新Kコース
CZ-653C（本体）…¥285,000
CZ-604D（0.31ステレオスピーカ付ディスプレー）…¥94,800
御希望ゲームソフト…¥サービス
合計¥379,800➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000PROⅡ X68000 新Lコース
CZ-653C（本体）…¥285,000
CZ-605D…¥115,000
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
合計¥418,000➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000PROⅡ X68000 Fコース
CZ-653C（本体）…¥285,000
CZ-602D（0.39カラーディスプレーテレビ）…¥99,800
住友3M 5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥402,800➡¥259,800

支払回数	ボーナス・頭金
¥7,200×48回	ボなし 頭なし
¥13,100×24回	ボなし 頭なし

一歩ふみこんだアートの世界 X68000 Rコース
CZ-604C（本体）…¥348,000
CZ-602D（0.39カラーディスプレー付ディスプレー）…¥99,800
住友3M2HD ブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥465,800➡現金大特価

支払回数	ボーナス	頭金
¥2,800×60回	ボ¥30,000	頭なし
¥5,000×36回	ボ¥40,000	頭なし
¥7,800×60回	ボなし	頭なし
¥11,900×24回	ボ¥30,000	頭なし

X68000PROⅡ X68000 新Jコース
CZ-604CTN（本体）…¥348,000
CZ-606TN（0.31カラーディスプレー）…¥84,600
CZ-8NJ2（インテリジェントスティック）…¥23,800
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥474,600➡現金大特価

支払回数	ボーナス	頭金
¥6,700×78回	ボなし	頭なし
¥8,000×60回	ボなし	頭なし
¥12,000×36回	ボなし	頭なし
¥17,300×24回	ボなし	頭なし

高解像度カラーで金や銀まで表現 X68000 新Sコース
CZ-604C（本体）…¥348,000
CZ-605D（0.39ステレオディスプレーテレビ）…¥115,000
住友3M2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥481,000➡現金大特価

支払回数	ボーナス	頭金
¥4,900×60回	ボ¥20,000	頭なし
¥7,300×36回	ボ¥30,000	頭なし
¥8,900×60回	ボなし	頭なし
¥17,800×24回	ボなし	頭なし

X68000PROⅡ X68000 新Pコース
CZ-652CBK（本体）…¥298,000
CZ-602D BK（0.39カラーディスプレーテレビ）…¥99,800
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥415,800➡¥260,000

支払回数	ボーナス	頭金
¥6,700×59回	ボ¥20,000	頭なし
¥8,000×60回	ボなし	頭なし
¥12,000×24回	ボ¥20,000	頭なし
¥17,300×18回	ボ¥30,000	頭なし

X68000をはじめソフト&周辺機器類は、当社池袋店・札幌店・旭川店・福岡店にて実演中です。各店X68000コーナーが常設されております。

X68000ソフト&周辺機器

品名	価格	品名	価格	品名	価格
SCSIボード（CZ-6BSI）	¥29,800➡現金特価	BF-68PRO	¥19,800➡¥16,800	Communication PRO-68K	¥19,800➡現金特価
システムサコムMIDIボード（SX-68M）	¥19,800➡¥15,300	CZ-6TU	¥33,100➡現金特価	Stationary PRO-68K	¥14,800➡現金特価
LANボード	¥268,000➡¥201,000	オムロンMD-24FP4Ⅱ	¥38,800➡現金特価	DATA PRO-68K	¥58,000➡¥43,500
RS-232Cケーブル（平行）	¥7,200➡現金特価	オムロンMD-24FP5Ⅱ	¥42,800➡¥33,000	BUSINESS PRO-68K	¥68,000➡¥51,000
RS-232Cケーブル（クロス）	¥7,200➡現金特価	ローランドMT-32	¥64,000➡¥54,400	NEW Printshop PRO-68K	¥19,800➡現金特価
インテリジェントコントローラ	¥23,800➡¥18,900	Hyperword	¥39,800➡現金特価	グラフィックライブラリ vol.1	¥8,800➡現金特価
トラックボール	¥13,800➡¥12,000	CYBERNOTE PRO68K	¥19,800➡現金特価	グラフィックライブラリ vol.2	¥8,800➡現金特価
ジョイカード（延長コード付）	¥3,200➡¥2,900	C compiler PRO-68K	¥44,800➡¥33,600	Musicstudio PRO-68K ver1.1	¥28,800➡¥21,600
CZ-8BSI（X-1用）	¥23,800➡現金特価	CARD PRO-68K	¥29,800➡現金特価	MUSIC PRO-68K〔MIDI〕	¥28,800➡現金特価
拡張I/Oボックス	¥88,000➡現金特価	CARD PRO システム手帳リフィル集	¥9,800➡現金特価	ソングライブラリ 101曲集	¥8,800➡現金特価
アンプ内蔵スピーカーシステム	¥36,600➡現金特価	CARD PRO 活用フォーム集	¥9,800➡現金特価	Sampling PRO-68K	¥17,800➡現金特価
システムラック	¥44,800➡¥35,800	SX-WINDOW ver1.0	¥6,800➡現金特価	SOUND PRO-68K	¥15,800➡現金特価

パソコン通信セット〔ソフトたーみのる2（¥12,800）+モデムMD24FP5Ⅱ2400ボモデム（¥42,800）〕➡¥42,800 通信セット〔ソフトX Talk-68K（¥12,800）+モデム MD12FS 1200ボモデム（¥21,000）〕➡¥27,300 X68000接続電子手帳セット〔ケーブル CE-300L（¥3,500）+サイバーノート68K（¥19,800）+電子手帳PA 8500（¥28,000）〕➡¥38,600 ★CU-21HD ステレオスピーカ付21インチディスプレー ¥148,000➡現金大特価 ★CZ-604D〔ステレオスピーカ付603Dディスプレー〕…➡現金大特価

最新お買得ソフトセット X68000 パーツセット1
シム・シティー…¥9,800
ナイトアームズ…¥9,700
サイバリオン…¥8,800
グラナダ…¥8,800
きゃんきゃんバニー…¥6,800
スターシップランデブー…¥8,800
定価合計¥51,700➡¥39,900
安すぎて表示できません クレジットでもお申し込み出来ます

ステレオMIDI音源セット X68000 パーツセット2
CZ-6BM1（MIDIボード）…¥26,800
CM-32L（ローランドMIDI音源）…¥69,000
MA-12AV×2（パワーアンプ内蔵スピーカー）…¥28,000
MUSIC-PRO-MIDI（ソフト音源）…¥28,800
ソングライブラリ（ソフト音源データ）…¥8,800
定価合計¥161,400➡¥134,900

支払回数	ボーナス・頭金
¥6,900×24回	ボなし 頭なし
¥12,800×12回	ボなし 頭なし

グレー限定お買徳セット X68000 新Tコース
CZ-652CGY（本体）…¥298,000
CZ-604DGY（0.31ステレオスピーカー付ディスプレー）…¥94,800
住友3M2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥410,800➡現金大特価

支払回数	ボーナス	頭金
¥5,400×72回	ボなし	頭なし
¥6,100×60回	ボなし	頭なし
¥9,100×36回	ボなし	頭なし
¥13,100×24回	ボなし	頭なし

グレー限定お買得セット X68000 新Uコース
CZ-603CGY（本体）…¥338,000
CZ-603DGY（カラーディスプレー）…¥84,000
ステレオスピーカー…¥10,800
住友3M2HDブランクディスケット…¥18,000
御希望ゲームソフト（上記ソフトよりお選び下さい）…¥サービス
定価合計¥450,800➡¥299,000

支払回数	ボーナス	頭金
¥6,100×72回	ボなし	頭なし
¥6,900×60回	ボなし	頭なし
¥10,300×36回	ボなし	頭なし
¥14,900×24回	ボなし	頭なし

X-68000 SUPER HD X68000 Oコース
CZ-623CTN（本体80M HDD内蔵）…¥498,000
CZ-606DTN（0.31カラーディスプレーチタン）…¥84,800
住友3M 2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥600,800➡現金大特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000 SUPER HD X68000 Iコース
CZ-623CTN（本体80MHDD内蔵）…¥498,000
CZ-602DBK…¥99,800
住友3M 2HDブランクディスケット…¥18,000
御希望ゲームソフト…¥サービス
定価合計¥615,800➡¥450,000

支払回数	ボーナス・頭金
¥10,300×60回	ボなし 頭なし
¥15,500×36回	ボなし 頭なし

全国出張サポート★
私共にてご購入いただいたX68000は全国出張サポートがうけられます。

チタンカラーも入ってますます充実 X68000 Hコース
CZ-604CTN（本体）…¥348,000
CZ-606DTN（0.31カラーディスプレー）…¥84,800
住友3M 5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥400,800➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000EXPERTⅡ X68000 新Cコース
CZ-603C（本体）…¥338,000
CZ-605D（0.39ステレオスピーカ付テレビ）…¥115,000
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
合計¥471,000➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000EXPERTⅡ X68000 新Rコース
CZ-603C（本体）…¥338,000
CZ-605D（0.39ステレオスピーカ付テレビ）…¥115,000
リップスティックアドベンチャー…¥6,800
ヴァリスⅡ…¥9,800
V'BALL…¥7,900
パワフル麻雀…
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト…サービス
合計¥503,300➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000PROⅡ X68000 Gコース
CZ-653C（本体）…¥285,000
CZ-603D（0.31カラーディスプレー）…¥84,000
住友3M 5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥387,800➡¥249,800

支払回数	ボーナス・頭金
¥6,900×48回	ボなし 頭なし
¥12,600×24回	ボなし 頭なし

X68000EXPERTⅡ X68000 新Dコース
CZ-603C（本体）…¥338,000
CZ-604D（0.31ステレオスピーカ付ディスプレー）…¥94,800
住友3M5'2HDブランクディスケット…¥18,000
御希望ゲームソフト…¥サービス
合計¥450,800➡現金特価
安すぎて表示できません クレジットでもお申し込み出来ます

X68000EXPERTⅡ X68000 Aコース
CZ-603C（本体）…¥338,000
CZ-603D…¥84,800
住友3M 5'2HDブランクディスケット…¥18,000
御希望ゲームソフト（人気ソフト上記よりお選び下さい）…¥サービス
定価合計¥440,800➡¥305,000

支払回数	ボーナス・頭金
¥8,300×48回	ボなし 頭なし
¥15,200×24回	ボなし 頭なし

ステレオMIDI音源セット・ホールラック X68000 パーツセット3
SX-68M（MIDIボード）…¥19,800
MT-32（ローランドMIDI音源）…¥64,000
MA-12AV×2（アンプ内蔵スピーカー）…¥28,000
CF-10（コンピュータミキサー）…¥22,000
PC-200（キーボード）…¥36,000
MU-1（ソフト音源）…¥19,800
ソングライブラリ（ソフト音源データ）…¥8,800
―お買上げのX68000と合せて御利用いただけます。―
定価合計¥198,400➡¥163,000

支払回数	ボーナス・頭金
¥8,300×24回	ボなし 頭なし
¥15,300×12回	ボなし 頭なし

カラーグラフィックセット X68000 パーツセット4
GT-1000（カラーハンディスキャナ）…¥79,800
≠5220（接続ケーブル）…¥7,500
Z's STAFF・PRO（グラフィック作成ソフト）…¥58,000
定価合計¥145,300➡¥103,800

支払回数	ボーナス・頭金
¥5,300×24回	ボなし 頭なし
¥9,800×12回	ボなし 頭なし

MUSIC PRO-68K ¥18,800➡現金特価 OS-9 X68000 ¥29,800➡¥22,300 Human68K ver2.0 ¥9,800➡現金特価 たーみのるⅡ ¥12,800➡現金特価 Z5 STAFF PRO-68K ¥19,800➡現金特価 C-TRACE ¥68,000➡¥38,000 Kamikaze ¥68,000➡¥49,600 Canvas PRO 68K（トローグラフィックツール）¥29,800➡現金特価 トローグラフィックライブラリVol.1 ¥9,800➡現金特価 トローグラフィックライブラリVol.2 ¥9,800➡現金特価

X68000シリーズ周辺機器

品名	価格	品名	価格	品名	価格
CZ-8NS1	¥188,000➡¥141,000	CZ-8PC5	¥94,800➡現金特価	I/Oデータ 2MB増設RAM	¥50,000➡¥36,500
CZ-6BN1	¥29,800➡現金特価	IO-735X	¥248,000➡現金特価	I/Oデータ 4MB増設RAM	¥88,000➡¥64,000
CZ-6VT1	¥69,800➡¥52,400	CZ-8PK10	¥97,800➡現金特価	GP-IBボード	¥59,800➡現金特価
CZ-6BV1	¥21,000➡現金特価	1MB増設RAM（CZ-600C専用）	¥35,000➡¥28,000	増設用RS-232Cボード	¥49,800➡現金特価
CZ-6PV1	¥198,000➡¥148,500	1MB増設RAM	¥28,000➡¥22,400	ユニバーサルI/Oボード	¥39,800➡現金特価
CZ-8PC3	¥65,800➡¥39,000	2MB増設RAM	¥79,800➡現金特価	数値演算プロセッサ	¥79,800➡現金特価
CZ-8PG1	¥130,000➡¥97,500	4MB増設RAM	¥138,000➡現金特価	FAXボード	¥79,800➡¥55,800
CZ-8PG2	¥160,000➡現金特価	I/Oデータ 1MB増設RAM	¥25,000➡¥18,000	MIDIボード	¥26,800➡現金特価

組合せ自由	激安金利にキャンパスクレジット	ゆっくり、お支払いは8ヵ月先から
各コース以外の組合せもコースをベースに周辺を合せたセットお支払いたって御希望のパターンをお組みいたします さあ、ご相談もお見積りも受注センターもしくは各店へお気軽に	手続きカンタン、大学生の為の超低金利クレジット 20歳以上の学生の方は原則として保証人様には連絡いたしません	クレジット業界最低の金利を有効に使って、支払いは最長8ヵ月後から始まるクレジットでも

ワールドインアオヤマの 9ハートサービス
実績のメンテナンス／完全バックアップシステム

クレジット
業 界一番のスーパークレジットで。
学 生の味方、キャンパスクレジットがますますワイドに。
お 支払いはナント！84回まで。
ゆ うゆうお支払いは8ヶ月先から。
サポート
万 一のときも完全バックアップ。
会員・保険
最 先端の"CLUB246"。
イ ンシュアランス・サービス
グ ーンとお得な下取りシステム。
ショップ
全 店統一のサービス 見て、触れて、納得。
B EST SHOPS BEST SERVISE。
サービス&ポリシー
グ ーンとお得な下取りシステム。

お求めは全国のAOYAMAへ

旭川店 ☎0166-25-7771
旭川市4条8丁目ツジビル
■営業時間 11:00～19:00

札幌店 ☎011-251-1777
札幌市中央区南2条西3丁目 リンクエギビル3F
■営業時間 11:00～19:30

札幌PAX ☎011-223-7771
札幌市中央区南2条西2丁目 ブロックビル6F
■営業時間 11:00～19:30

池袋ソフト店 ☎03-989-1171
豊島区東池袋1-28-6 パールシティビル2F
■営業時間 11:00～19:00

池袋ショールーム ☎03-989-1123
豊島区東池袋1-27-12明治生命池袋ビル9F・首都高速道路側
■営業時間 11:00～19:00

福岡ユーテック店 ☎092-716-7771
福岡市中央区渡辺通り4-9-25 ユーテックプラザ4F・地下鉄天神駅下車3分
■営業時間 11:00～19:30

●配送日指定OK●土・日配送OK●札幌BLOCKに買取専門ショップOPEN●専門スタッフが常時待機●中古パソコン4ヶ月間保証●FAX見積りサービス●もちろんクレジットカードも使えます●カード払いでキャッシュレス

ワールド イン アオヤマにおまかせください。

返品・交換について
返品・交換は商品到着後3日以内にお願いいたします。
〈返品・交換をお受けする場合〉
●不良品（キズ・汚損）が届いた場合
●ご注文商品と違う商品が届いた場合
上記の場合、返送料は当社が負担いたします。ただしお客様の事由（注文ミスなどによる返品の場合、返送料はお客様の負担となりますのでご了承下さい。
〈次の場合返品・交換はお受けできませんのでご注意下さい〉
1 1度ご使用になった商品 2 お客様の責任でキズや汚れが生じた商品 3 全ソフト商品

日中ご不在がちな方はお振込
昼間ご不在がちの方、お急ぎの方は銀行振込が便利です。
お申し込みからお届けまで、お時間をとりません。
●銀行振込みの場合

取引銀行	住友銀行 池袋支店
口座番号	普通 1065392
口座名	株式会社 ワールドイン アオヤマ

返品・交換についてのお問い合わせは お客様相談室 03-987-7795

電話でのご注文の場合
☎ 03-987-7771
お電話番号はおかけ間違いのないようにお願いします。
北海道受注センター ☎011-251-6771
九州受注センター ☎092-672-7771
お好きな時間にお電話を！
テレフォンショッピング
電話受付時間 ■月曜日～金曜日 10:30～20:30 ■土・日曜日・祭日 10:30～19:00

クレジットカードもOK！
クレジットカードをお持ちの方お支払は1回払いです。お申し込みの際、①カード名②会員No.③有効期限をご連絡下さい。
カードでお申し込みの場合、販売価格が変わりますのでお電話にてお問い合せ下さい。

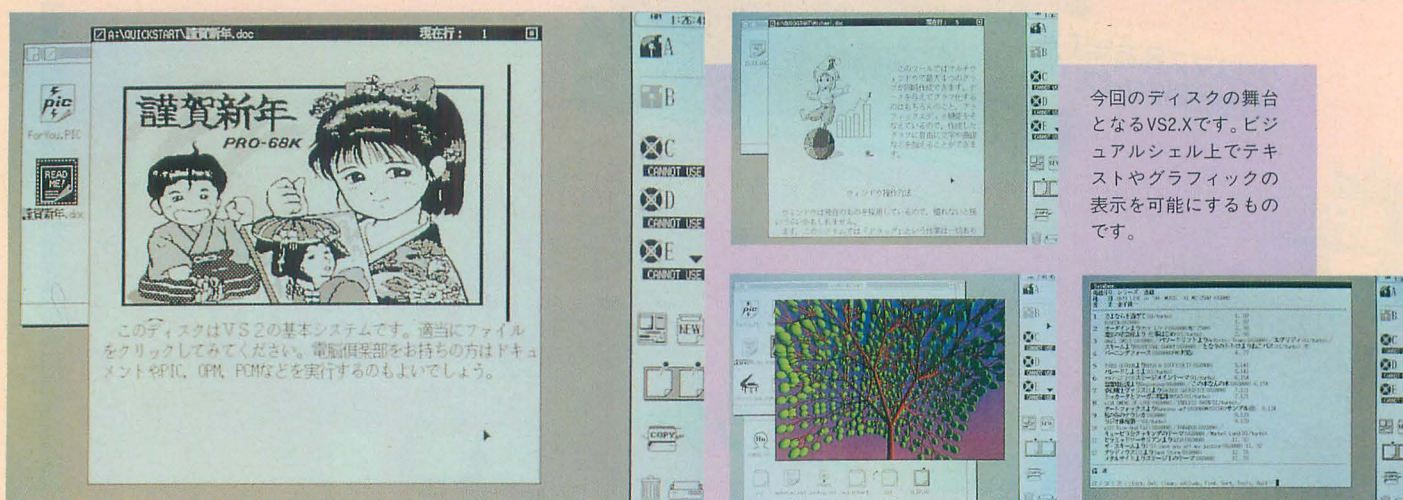
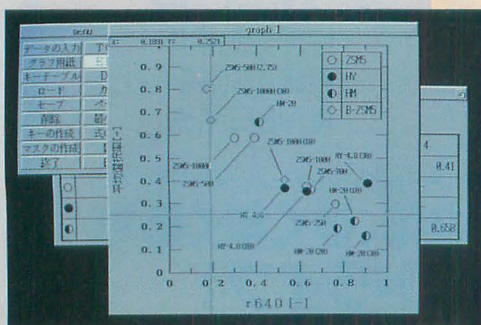
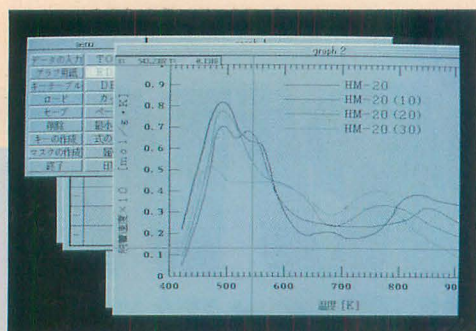
ファクシミリでご利用の場合
03-985-522
●ご注文方法（黒色のボールペンまたはサインペンでご記入下さい）
1 電話番号・住所・氏名又はお…号、お支払い方法をご記入下

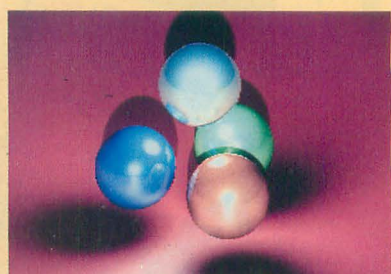
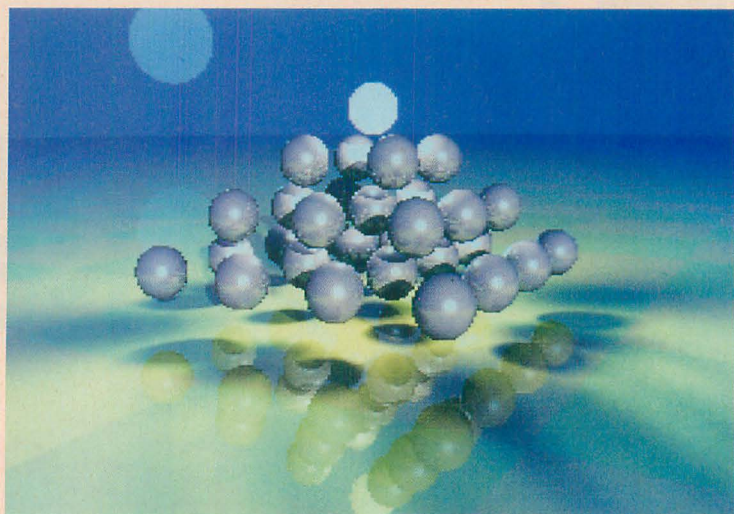
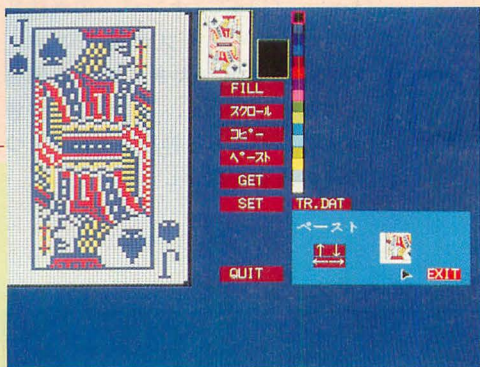
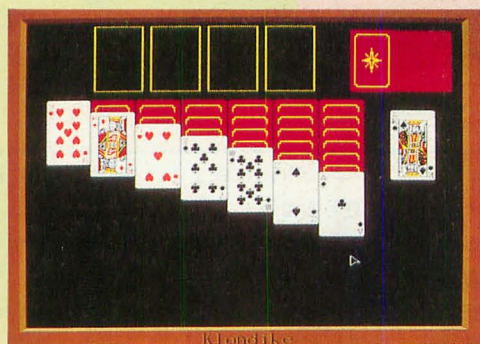
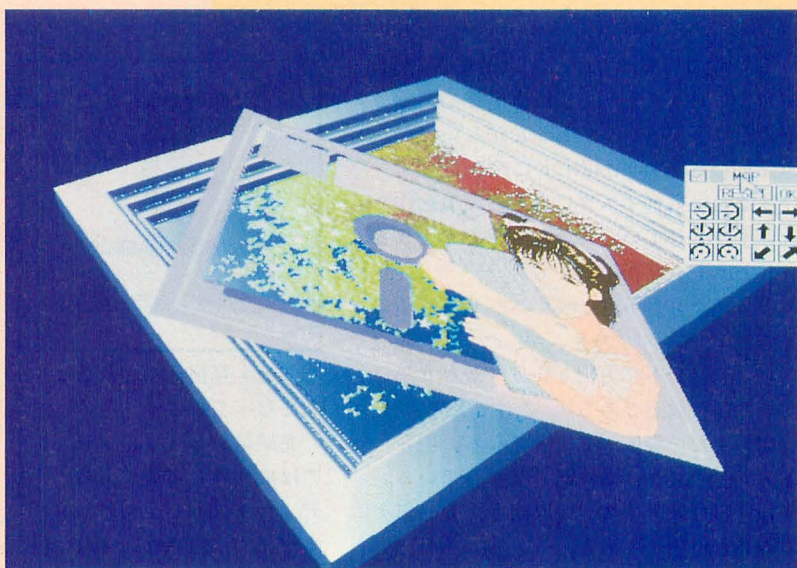
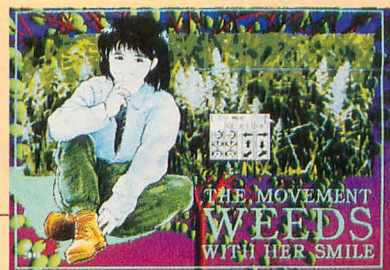
お客様相談室
03-987-779
すでにご注文いただいているお届け時間（時期）、メンテナンス、そのお問い合せは上記へお電話下さい。

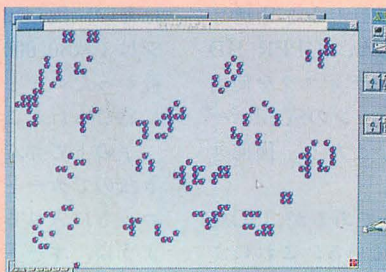
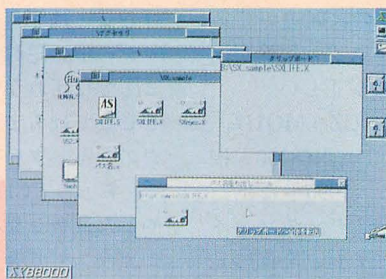
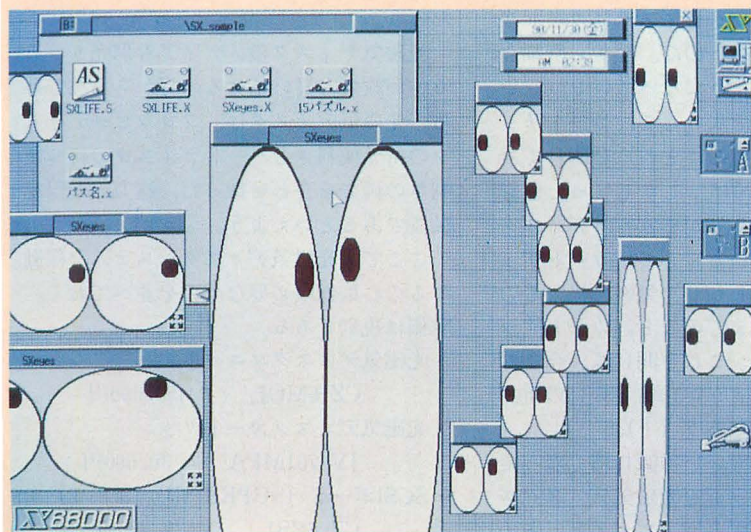
サポート
〈万一のときにも完全バックアップ〉
万一の初期不良があった場合では万全の体制でお客様をフォローします。通常の初期不良ワクをもうけ、最長1ヶ月間まで、新品との交換しております

謹賀新年PRO-68K

あけましておめでとうございます。では，いきなり付録ディスクのお披露目といきましょう。

これが今回のディスクに詰め込まれたプログラムの数々です。詳しい解説は本文をどうぞ。

今回のディスクの舞台となるVS2.Xです。ビジュアルシェル上でテキストやグラフィックの表示を可能にするものです。

グラフ作成支援ツールMichael。マルチウィンドウで４枚までのグラフを扱えます。簡易グラフィックエディタ機能つき。

柔らかな影を持つレイトレーシングツールHASHによる画像の作成例。面光源からの光が独特の雰囲気をかもしだしています。

コンパイラ対応のCARD2.FNC。これはカードエディタの画面。256色データにも対応しました。花札だろうが，モンメカだろうが，なんでもこなせます。

カードゲームのサンプルはKLONDIKE。有名なトランプのひとり遊びです。ルールがわかるまでは手こずりますが，かなりハマれるゲームです。

Z'sSTAFFに拡張機能を！ 1990年8月号で扱ったランダムフラクタル処理，透過光のようなフレア処理，3次元マッピングなどの例。

ディスクに含まれるSX-WINDOW用アクセサリ。X-WINDOWでお馴染みの目玉からライフゲーム，パス名取り込みツールなど。

製品紹介

MOの時代がやってきた

光磁気ディスクシステムCZ-6MO1

Tan Akihiko 丹 明彦

シャープ純正CZナンバーの光磁気ディスクドライブが発売され，X68000の環境が大きく買える価格ではないとはいえ，ここはひとつ皆さんのふところと神経を逆撫でしつつその概要を紹介しよう。

僕は現在，編集部マシン室のX68000 SUPER-HDをほとんど私物化している。その僕は今，嬉しくてしかたがないのだ。SUPER-HDの80Mバイトに飽き足らず，今度は光磁気ディスクに自分の環境を作り上げようとしているのだから。そうできる日がとうとうきたのだから。

ディスク容量不足に嘆くユーザー待望の光磁気ディスクシステムが，ついに[1]発売された。しかも45万円という，かなり思い切った価格設定である[2]。これでX68000の環境がまた一歩広がることになる。現在，光磁気ディスクは編集部で稼働している。今月号の付録ディスクの原版も，光磁気ディスクに格納してある。

■製品概要

すでにご存じのとおり，光磁気ディスクは，パーソナルコンピュータなどの大容量外部記憶装置のひとつである。

原理的にはハードディスクよりもむしろCD-ROMに近いのだが，使う上では，大容量のハードディスクという感覚である。なぜなら，光磁気ディスクは，読み出しだけでなく書き込みもできるからである。光磁気ディスクは，この点でCD-ROMよりも大きくポイントを稼いだ。ただ，これでCD-ROMが劣っているというのは間違いである。CD-ROMと光磁気ディスクは用途が違うというふうにとらえるのが正しい。

光磁気ディスクユニットCZ-6MO1は，SCSIインタフェイスを介してX68000と接続するようになっている。1990年に出たX68000SUPER-HDはSCSIインタフェイスを標準装備している。これは，今回の光磁気ディスクもそうだが，これからの外部記憶の標準はSCSIになるだろうと見越してのことであろう。もちろん，SUPER-HD以外のユーザーでも光磁気ディスクを使用することはできる。オプションのSCSIボードを利用すればよいのだ。ただし，後部スロットがひとつつぶれる。

次に，記録フォーマットであるが，ISO規格に準拠したものとなっている。これにより，他と互換性がとれるようになっている。

光磁気ディスクのメディアは取り外し可能である。ディスクはガラス製で，さすがにCD-ROMのように丸裸とはいかず，3.5インチディスクを思わせるプラスチック製のカートリッジの中に収まっている。カートリッジは，音楽用CDのケースくらいの大きさである（光磁気ディスクは5.25インチ)。カートリッジ表面をしげしげと眺める。A面やB面などとある。そう，光磁気ディスクは両面使えるのだ。よく見ると，ライトプロテクトタブもついている。ライトプロテクトのかかるハードディスクはそれほどないことを考えると，これは実にありがたいことである。

そのままでは残念ながらディスクの姿を拝むことはできない。ディスクはカートリッジの中に隠れているのだ。だから些細なことでは汚れないようになっている。しかしここではちょっと失礼して，カートリッジのシャッター（窓のことね）を開けてみる。ちょっとCDに似た，しかしずっと色鮮やかな虹色を見ることができる。ディスクがガラスでできているにしてはカートリッジは意外に軽い。しかし，気軽に扱えるのはその値段を知るまでのこと。カートリッジは1枚30,000円。これを聞けば，今まで軽かったカートリッジが10倍くらいの重さに感じられることだろう（！）。

手の上で弄ぶのもいい加減に怖くなってきたのでカートリッジを光磁気ディスクユニットに入れる。すると，ビデオデッキのように，オートフロントローディングをしてユニット内に収まる。少し（数秒）待つと使えるようになる。取り出すときはイジェクトボタンを押す。若干の時間をおいて（処理の後始末をしているのだろう)，やはりビデオデッキのようにカートリッジが吐き出されてくる。

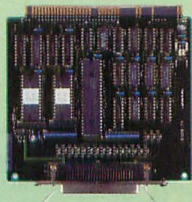
最大のウリであるディスクの記憶容量であるが，両面で594Mバイト。一度に使えるのは片面だけ[3]なので，実質300Mバイト弱だが，それでも十分すぎる容量とはいえるだろう。書き換え回数100万回以上（つまり，めったなことでは限界に達しない)。これで1枚30,000円なら十分に安いといえるだろう。

光磁気ディスクのメディアも別売りである。何枚でも買い足せるのだから当然である。光磁気ディスクは，フロッピーディスクの持つ便利さとハードディスクの持つ便利さの両方を合わせ持った，魅力的な記憶装置であるといえよう。

ここで，光磁気ディスクシステムを稼働するのに最低限必要なものを並べておく。定価は税別である。

・光磁気ディスクユニット
　CZ-6MO1　450,000円
・光磁気ディスクカートリッジ
　JY-701MPA　30,000円
・SCSIボード（SUPER-HDには不要）
　CZ-6BS1　29,800円

当然，OSもSCSI対応にしなくてはなら

SCSIボード CZ-6BS1について

光磁気ディスクと同時に，それをSUPER-HD以外のX68000と接続するためのSCSIインタフェイスボードもシャープから発売された。

SCSIとはSmall Computer System Interfaceの略で，パーソナルコンピュータと周辺装置との間でデータをやりとりするための標準規格である。その周辺装置は，主に大容量ハードディスクや今回紹介した光磁気ディスクといった外部記憶装置である。

X68000のSCSIインタフェイスは，SCSI規格の外部記憶装置を7台までディジーチェーン接続することができる。現在，編集室のSUPER-HDにも内蔵ハードディスクと光磁気ディスクの合計2台のSCSI装置が接続されている。

SUPER-HD以外のX68000で光磁気ディスクやSCSIハードディスクなどを使いたい場合は，このCZ-6BS1が必要である。そのためのSCSIドライバやフォーマットを行うソフトウェアは付属している。

ないが，これはSCSIボードにソフトウェアが同梱となっているので，特別に揃える必要はない。

■使ってみる

一般に光磁気ディスクをHuman68kで扱うためには，SCSIデバイスドライバであるSCSIDRV.SYSを組み込んでおく(CONFIG.SYSに書いておく）必要がある。これを忘れると，せっかくつないだドライブが認識されないという間抜けな事態に陥るので注意。

初期設定は通常のハードディスクと同じ。まず初期化および領域確保をして，300Mバイト近い領域にいくつかのパーティションを切る。1面まるごと1ドライブというのも，豪快でいいかもしれない。

使ってみるといっても，より大容量になったハードディスクという以外の感想はない。ただ，この大容量というのがとてつもなくありがたいのだ。画像ファイルをたくさん転がしておくのは，たとえハードディスクといえども犯罪行為に近いものがある。その点光磁気ディスクなら，巨大な倉庫として使うこともできる。PICファイルを収集して1カ所に置いておくのもよし，サンプリング音をまとめてライブラリにしてしまうもよし。ハードディスクでもかなりつらいTeXのファイルをしまい込んだところで，痛くもかゆくもない。しかも，これらがいつでも利用したいときに素早く利用できる。まったくこたえられない。

速度の点はやや心配されていたが，使っている上では特に遅いと感じない。十分にハードディスクの代わりができることと思う。具体的な数字を挙げておくならば，

- 平均データ転送速度（読み出し時）：
  635Kバイト/sec
- 平均シークタイム：
  105ms

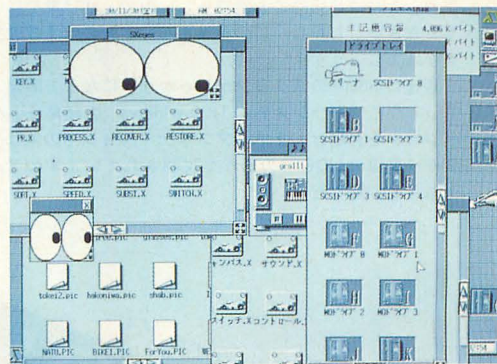
ということである。単純な数値比較はできないが，ハードディスクと比べてもそれほど遜色ないはずだ。実際，いろいろなプログラムで試してみると，確かに多少余計に時間はかかるが，その差はほんのわずかなものである。

動作音は，ハードディスクに比べると若干気になる。それでも，ソニーの光磁気ディスクに比べると相当静かだというのは，ソニーの光磁気ディスクドライブを実際に使っている祝一平氏の弁。もちろん，昔のハードディスクに比べるとはるかに静かである。

SX-WINDOWは光磁気ディスクにしっかり対応していて，ドライブトレイには「MOドライブ」の文字が出てくる。MOとはMagneto Optical diskの略で，要するに光磁気ディスクのことを指す。逆に，ビジュアルシェル（バージョン2）は光磁気ディスクだけでなくSCSIデバイス全般を認識しない[4]。

画像が大量に置けるのは嬉しいことである。CGをやっていると，なにかと画像用のファイルがたくさんできる。出来上がった画像はPICで圧縮してしまえばまだいい。しかし，マッピングに使うような大きくてなおかつ圧縮のできない(圧縮するわけにはいかない）画像ファイルはそうそう長いこと転がしておくわけにもいかない。100Mバイト単位で気兼ねせずに使えるというのは，まさに夢のような環境。通信で絵や音楽のデータをいっぱい抱え込んでいる人にもおいしい。ほしいデータをフロッピーをとっかえひっかえして探す憂鬱から解放されるのだから。

だが，画像をもっと大量に扱っていると，光磁気ディスクといえどもさすがにあふれることもあり得る。とくにアニメーションをするようになったらそうなる可能性は大きい。そのときは新しくメディアを買うのだ。1作品1枚などというのも辞さない。フロッピー600枚に匹敵する容量のディスクが，フロッピー300枚ぶんの値段で手に入ると思えば安いもの。しかもパフォーマンスはフロッピーをはるかに上回る。

ちょいと初期投資は大きいが，後々までおいしく使える光磁気ディスク。シャープ純正だからX68000にも安心して使える。今の環境でハードディスクが狭いとお感じの方は，住み替えを真剣に検討すべきなのかもしれない。

1）　シャープは光磁気ディスクに関してはかなり初期に手をつけていたほうで，この製品も実は最初のものではない。ただしシャープが以前に作った光磁気ディスクシステムは実に160万円もするしろものであったが……。

2）　45万円という価格は，1年半ほど前にソニーが達成（？）している。しかしいずれにしても，価格設定としてはかなりきついほうで，採算ラインには乗らないという話も聞く。

3）　ディスクを入れ替えたり裏返したりすれば，ドライブの設定状態（つまり，AとBドライブがハードディスクでCが光磁気ディスク，といったような）がまったく変わってしまうので，X68000本体をリセットして，システムを立ち上げ直さなくてはならない。形がフロッピーディスクに似ているし，イジェクトもできるのだから，これはなにか不便なような気もする。しかし，光磁気ディスクの場合，メディアの入れ替えはハードディスクを繋ぎ変えるのとおなじくらいおおごとなのだから，リセットすることは必要である。第一，光磁気ディスクはフロッピーと違い，パーティションを切って複数のドライブに分けて使うようになっている（ハードディスクと同じ)。入れ替えたディスクのパーティションが前と異なれば，アクセスに破綻をきたすのは目に見えている。

しかし，欲をいえば，レーザーディスクのように両面連続使用をしたり，両方の面のドライブを同時にアクセスできたりするともっといいのかも。

4）　というわけで，X68000SUPERにはビジュアルシェルは付属していない。が，今月号の付録ディスクにはVS.Xとそれを拡張するVS2.Xが収録されている(もちろんSCSIにも対応している）のでSUPERのユーザーの方も試していただきたい。

Oh!X Graphic Gallery

DōGA・CGアニメーション講座

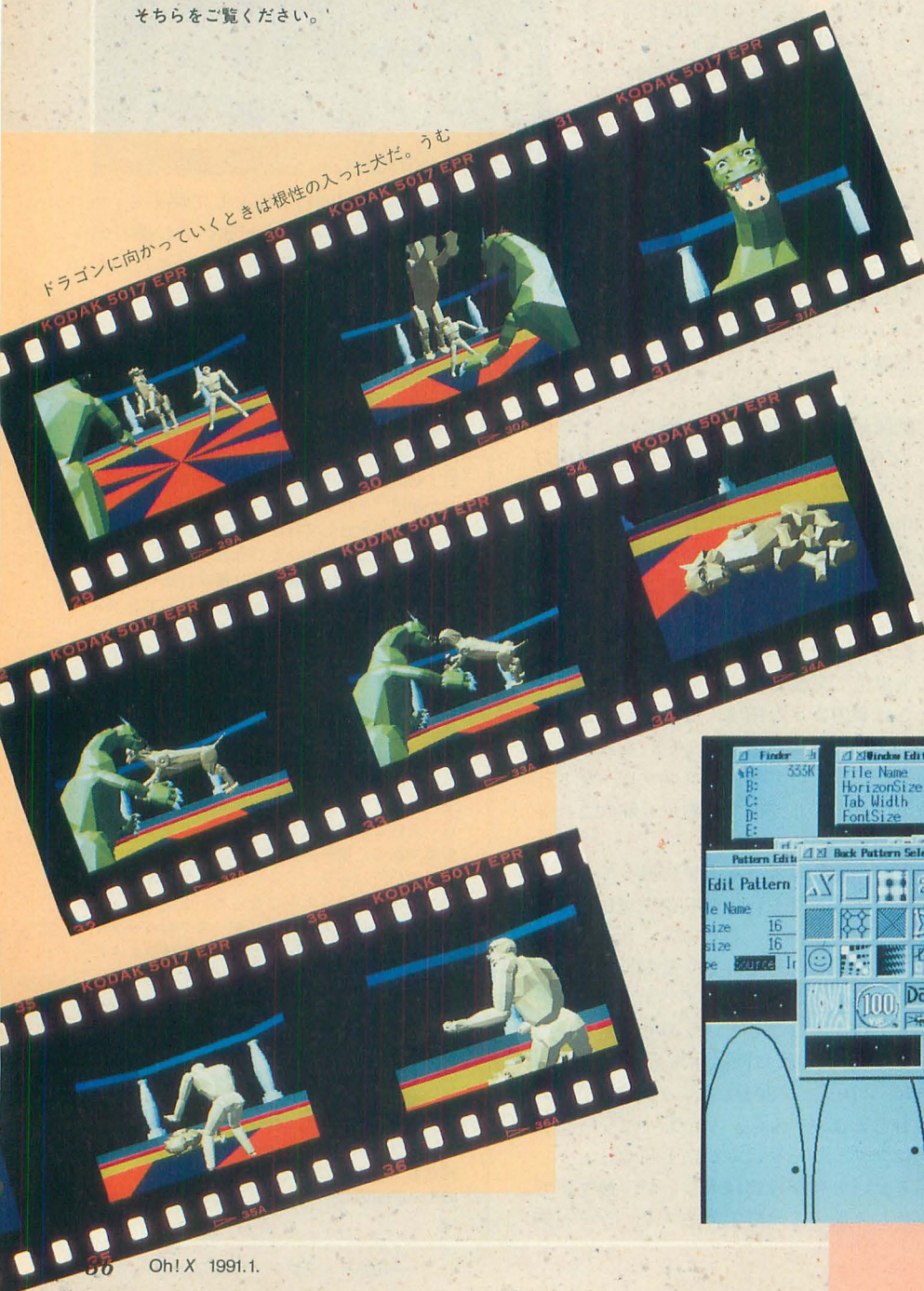
今月はアニメーション2作品と，DōGA期待のウィンドウシステム「KO-Window」の発表です。

まずは，モデラー高津の今月のアップデータです。これは，もしやあの“ウルト○ホーク1号”では？　なんでもちゃんと3機に分かれるそうなんです。いやあ，凝ってますねぇ。

お次は本職の画家の方が作ったという「SWORD」です。なんとディスク5枚組だったというからすごい大作ですね。これは技術的にも素晴らしいので，ぜひ参考にしてください。しかし，この画家さん，本職のほうは大丈夫なんでしょうか？

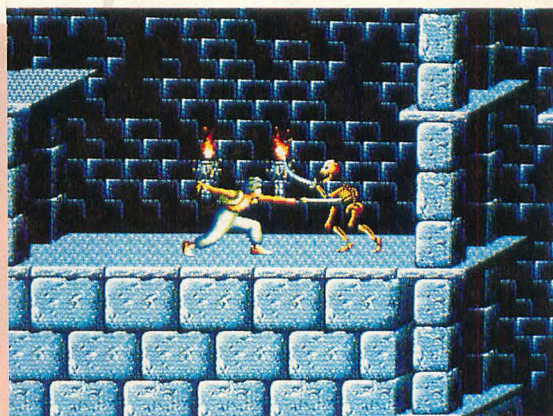
最後にDōGAのスタッフがしゃかりきになって開発しているウインドウシステム「KO-Window」の開発途中バージョンをお届けします。アプリケーションその他はモデラー高津のLOGINのほうで説明されていますので，そちらをご覧ください。

前回に続き飛行機シリーズ第2弾ってとこかな

ドラゴンに向かっていくときは根性の入った犬だ。うむ

ただいま開発中のDōGAオリジナルウィンドウシステム。CGAシステムVer 3は，このKO-Window上で動く予定とのこと。

プリンス・オブ・ペルシャ
アメリカで大人気のアクションゲーム。ワナでいっぱいの迷宮を関節くねくねの少年を操作し，1時間以内に姫を救いだせ！

SOFTWARE information

あけましておめでとうございます。こうして1年を振り返ってみると，いやー，ゲームって増えましたねぇ。てなわけで，このページも今月号から内容が少し変わります。ぜひ感想を聞かせてね。

話題のソフトウェア

今月のトップはこれ，ブロダーバンド・ジャパンの**プリンス・オブ・ペルシャ**です。このゲームはアメリカ産のアクションアドベンチャーで，本当に滑らかな動きと簡単操作，独特のグラフィックがウリ。X68000版ならグラフィックは当然。AMIGA版で出力されているようにリアルな効果音も，ぜひ再現してもらいたいところです。発売日は未定だけど楽しみに待っていようね。

移植ものといえばこちらもそう，ウェーブトレインの**スペース・ローグ**。これはAPPLE IIで人気だった同ソフトの移植もの。一見，宇宙を舞台にしたフライトシミュレータなんだけど，宇宙基地につくと画面がガラッと変わってRPGに。まさに一粒で二度おいしいってヤツですね（ちょっと例が古かったわね……)。これはすでに発売中。

さて，次はヘルツの**ダイナマイト・デューク**にいきましょう。アーケードゲームの移植とあって，以前とは勝手が違うのか発売がやや遅れていましたが，とうとうお目見えです。アクションとシューティングの要素を持つこのゲーム，一見の価値はありそうですね。こちらも発売中です。

さあ，XIturboユーザーの皆さん，お待たせしました。やっとあの**プール・オブ・レイディアンス**が発売されました。AD&D®の“フォーゴトン・レルム”を舞台にした本格ファンタジーRPG。設定からキャラクターまで凝りに凝ったぶんだけ発売は遅れちゃったけど，素直に喜べるぞ。わぁい。

シューティングゲーム，猛追!!

1	ラグーン	1
2	ナイアス	一初
3	パロディウスだ！	8↑
4	ダンジョン・マスター	6↑
5	シムシティー	2↓
6	ソーサリアン	4↓
7	遊撃王II～エアー・コンバット～	7
8	ポピュラス	5↓
9	サイバリオン	3↓
10	イメージファイト	一初

2カ月連続でトップはラグーン。しかし，今月の話題はそんなことではない。

なんとナイアスが初登場2位!!

あのジェノサイドと同様のスタートをきったというから驚き。さっそくハガキの声を聞いてみましょう。

「3面のラスタースクロールは凄い」「シューティングの特訓になる」「ゲームバランスがとれている」「ラスターが凄い」「グラフィックがハデだから」「ラスタースクロールがイカす」……なんかラスタースクロールの話ばっかり（数えてみたら約3割)。ま，新規参入なのに技術力がしっかりしてるというコトの表れなんでしょうけど。このハガキの熱意からすると，来月はひょっとしてひょっとするかもしれないぞ。さあ大変だズームファン。

ナイアスだけでなく，パロディウスだ！，イメージファイトなどほかのシューテイングも軒並みランクアップ。前評判だけで3位のパロディウスだ！ は，X68000とグラディウスシリーズとの因縁もあって，デキ次第ではアフターバーナー並みのヒットもありえそう。イメージファイトはゲーセンで難しさにあきらめた人が再びチャレンジしているケースが多いようです。

アクション以外ではダンジョン・マスターが唯一ランクアップ。登場時期のわりに健闘しています。なお，「カオスの逆襲」は別集計なのでハガキにはサブタイトルまで書いてください。よろしく。

10位のすぐ下ではジェミニウイング，銀河英雄伝説IIなどが頑張っています。この強力なメンバーによるバトルは，来月に向けてさらに激しさを増す模様。'91年最初のランキングに乞うご期待！　　(浦)

パロディウスだ!

早くもファンから熱い期待が寄せられているパロディウスだ!，今回も画面写真しか間に合わなかったのでこれで我慢してちょうだい。

しかし，画面を見るかぎりホントに完璧なデキ。こころもち横幅がつまってるかなという気もするけど，でも本当にX68000の画面なのかどうか疑っちゃうくらいだ。ガニ股歩きのおネエさんもモアイ戦艦もそのまんま。ファンキーなグラフィックも再現されていて，このままスイスイ動いちゃったら，X68000の限界をまたまた塗り替えるとんでもない作品になっちゃうかもしれない。

とにかく，大型きょう体を除いたら去年ゲームセンターで注目度No.1だったソフトだけにそのデキには注目度大なのであります。　(浦)

X68000用　5″2HD版　価格未定
コナミ　☎03(262)9110

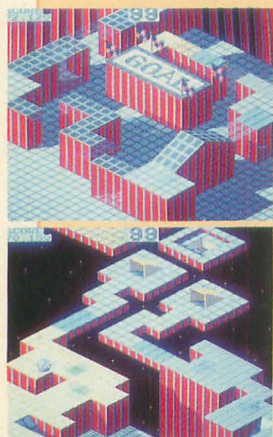
＊画面は開発中のものです。

エメラルドドラゴン

みんなが待っていたエメラルドドラゴン，そう，あれほど待ったエメドラもとうとう発売のときがやってきたのだよ。このエメラルドドラゴンというゲーム，プレイヤーは主人公のアトルシャンになって4人の仲間たちと聖地イシュバーンをさまよい戦っていくというフィールドタイプのRPGなのだ。

とにかくこのエメラルドドラゴン，マップが広い，話も飽きさせない展開でぐいぐい引き込む，アニメ絵のタムリンがかあいいかあいい，本当にかあいいなど，先に発売になったPC-8801ユーザーの間では，寝ても醒めてもエメドラ，のエメドラゾンビまで出てしまったというくらい(?)とんでもなく評判のよかったゲームなのだ。でもってX68000版は，それよりさらにさらにグラフィックは見目麗しき水彩調になり，音楽もX68000の音源に合わせてバリバリにパワーアップされているぞ。(で)

X68000用　5″2HD版6枚組　9,800円(税別)
グローディア　☎03(220)5226

マーブルマッドネス

＊画面は開発中のものです。

ホームデータの手によって移植が進んでいる「マーブルマッドネス」の画面写真が届いたのでご紹介しよう。基本部分はもう完成しているようで，これから敵キャラやいろんな仕掛けを組み込むようだ。

マーブルマッドネスのミソはこの仕掛け。基本は，玉を転がして時間内にゴールまで運んでやるというものなんだけど，ハネ橋や波打つ床，突然現れて玉を吸い込もうとする掃除機まで，あの手この手で行く手を阻んでくるのだ。

このゲーム，編集室ではAMIGA版ですでに遊ばれ尽していて，わざわざトラックボールまで買ってくるほどの人気。画面を見るかぎり，移植はなかなか順調のようだから，X68000ユーザーが手にする日もそう遠くない。完成版が届き次第真っ先に紹介するのでお楽しみに。　(浦)

X68000用　5″2HD版　価格未定
ホームデータ　☎078(261)2790

ザ・スーパーラスベガスII

春の旅行にはアメリカに行って，ついでにラスベガスで大金持ちになっちゃおうかなどと思っている読者もおそらく5000万人(推定)はいるかと思いますが，世の中そんなに甘くない。やはり何事も傾向と対策が肝心。まずこのザ・スーパーラスベガス2で鍛えましょう。

ポーカー，セブンブリッジなどのカードゲームはもちろん，バックギャモンやルーレット，なぜかチンチロリンまであります。ひとつのゲームで勝った分はほかのゲームに持っていくことができるので，ちょっと負けてもあきらめないでほかのゲームに挑戦してみるのもいいんではないでしょうか。グラフィックがなかなかアメリカンしていてよいですぞ。ただラスベガスのわりにはスロットマシンがないけど。

さあ，これさえあればあなたも明日から大金持ち気分。毎日かけそばのかわりにきつねそばが喰えますぞい。　(浦)

X68000用　5″2HD版2枚組　12,800円(税別)
日本デクスタ　☎03(839)4711

アステロイド・クイーン/JANJON デルタアーム/ニニンバトル

タケルを通じて販売されているログイン誌の「ソフトウェア・グランプリ」入賞作から，最近発売されたものをいくつかご紹介しよう。

まずグランプリ受賞作，アステロイド・クイーン。マウスで自機を操作して小惑星を突き進み，敵UFOの母船を破壊するというゲーム。マウスによる操作感覚がウリだとか。

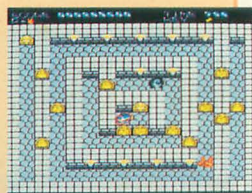
JANJONはアクションパズルゲーム。じゃんけんがモチーフで，敵も自分も手の形をしているのだ。自分の形を変えてしまうアイテムや，逆に特定の形でないと通れないドアもあったりして，面をクリアするのに相当悩みそう。

デルタアームは横スクロール型シューティング。しかし，自分でパワーアップアイテムを出せたり，やられ続けていると難易度がさがったりとなかなか斬新な試みが投入されている。ワンパターン化が叫ばれるシューティングゲームに一石を投じる作品となるか?

最後のニニンバトルはジョイスティック2本をつなぎ，2人で対戦するアクションゲーム。武器や能力の違うロボットのなかから好きなものを選んで戦うことができる。

いずれもアマチュアの作品だが，完成度はなかなか。2,000円という低価格もうれしい。

アステロイド・クイーン　X68000用　5″2HD版
JANJON　X68000用　5″2HD版
デルタアーム　X68000用　5″2HD版3枚組
ニニンバトル　X68000用　5″2HD版
以上すべて2,000円(税込)
ブラザー工業(タケル)　☎052(824)2493

アステロイド・クイーン

JANJON

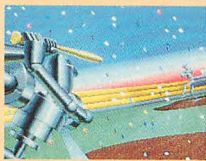
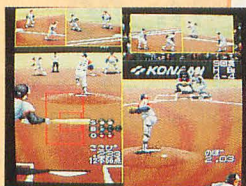
デルタアーム

ニニンバトル

生中継68

コナミの生中継68の写真が大量に届きました。このゲーム，とにかく画面へのこだわりがすごい。バックネット，一塁三塁スタンドの構図は当たり前。ライト，センター，レフトスタンドの画面もあるし，風船飛ばしやプロ野球ニュースとそのオープニング（！）まであるんです。これだけ大量の写真を送りたくなる気持ちもわかろうというもの。キャラクターはサイズも動きのハデさもバッチリみたいだし，２人プレイ時に専用の画面が用意されている(とおぼしき)写真もあって，熱意のほどがうかがえます。

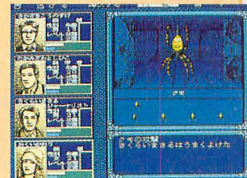
X68000用野球ゲームも再び元気を取り戻したようで嬉しい限り。この生中継68が，世間にコナミとX68000の力をガツンと見せつける一作になることを期待したいですね。　(浦)

X68000用　5″2HD版　価格未定
コナミ　☎03(262)9110
＊画面は開発中のものです。

ラプラスの魔

X1やPC-8801用などで先に出ていたので名前ぐらいは知ってるって人も多いラプラスの魔，移植にあたってちゃあーんとＸ68000のパワーに対応してオペレーション・ビジュアル・ミュージックすべてがリニューアルされているのだ。

このラプラスの魔はちょっと変わったタイプのRPG。ファンタジータイプのゲームとはちょっと違って，敵への攻撃に魔法ではなく精神攻撃が使えたりとか，金を稼ぐのにモンスターの写真を撮って街で売ったりと，なかなかにユニークなゲームシナリオなのだ。そうそう，クトルー神話との関わりも深いという噂だから神話オタッキーにはこたえられないかもしれない!?

現在のところ制作も順調。来月は完成版でのレビューができると思うのでお楽しみにっ！　(……に，してもロードス島はどこいったんだ，ロードス島は!?)　(で)

X68000用　5″2HD版3枚組　価格未定
エム・エー・シー ハミングバードソフト☎06(315)8255

中華大仙

アーケードで出ていた，TAITOの「中華大仙」がX68000で発売されることになりました。移植を担当したのはおなじみSPSということで，出来のほうはまず心配無用。このゲームはその名から察せられるとおり中国が舞台。中国というと西遊記。西遊記というと孫悟空。というわけで，この「中華大仙」は孫悟空が主人公の横スクロールシューティングゲーム。迫りくるのは，ぎょうざ，シュウマイなどいかにも中華的な敵キャラ。背景は水墨画風。それに加えて中華風の音楽が奏でられるとなると，これはもう中国人になりきって(？)バシバシと撃ちまくるしかない。また，ボスキャラを倒しステージクリアすると，「大きな仙人」がほめてくれる。うーん，とっても「中華大仙」。　(R.A.)

X68000用　5″2HD版　価格未定
シャープ　☎03(260)1161

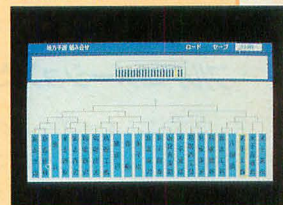
栄冠は君に

高校野球を舞台に監督として手腕をふるうシミュレーションゲーム「栄冠は君に」が完成。

全国の高校を収録し強さもだいたい似せてあるというヘビーそうなゲームだが，マウス操作のおかげでその辺の重さは感じさせずに自然にプレイできる。

でも，なにはともあれまず練習。部員の中から正選手，主将を選び，全国大会を目指して地道に練習を重ねていくのだ。１人ひとりに特別メニューを組むこともできるし，全員に講習を受けさせたりもできる。とにかくここでは地道な努力（なにしろ練習期間が40日！）が必要だ。しかし，ここでちゃんと頑張った人には，必ず勝利の女神が微笑むはず。

さあ，甲子園優勝を目指してガンバ！　(浦)

X68000用　5″2HD版3枚組　9,500円(税別)
アートディンク　☎0474(77)7541

あのゲームミュージックフェスティバルが，CD/ビデオになったって!?　そりゃ買わなきゃ，すぐ買わなきゃ。はへはへ

※ZUNTATAはTAITO，S.S.T.はSEGAのゲームミュージックスタッフの名称です。1990年８月25日にこの２大ゲームメーカーによるライブが行われたのでした。

●ZUNTATA LIVE

ゲームミュージックのようなインスト（歌の入らない楽曲）のライブ演奏は大変だ。ボーカルのある曲なら視聴者は歌を追いかけて聴くため演奏者のミスは耳につかないものだが，インストの場合は楽器と楽器の音色のからみの心地よさを楽しむ音楽のため，ひとつのトチリが大失敗につながることさえある。ZUNTATAはライブ演奏は初めてとのこと。確かに表情は緊張を隠せてなかったけど演奏はさすが，うまいうまい。選曲もいいし，アレンジもいい。「DADDY」のキーボードソロご苦労様でした。こうして聴いてみるとTAITOのゲームミュージックって作りが実にライブ向きですねぇ。

●S.S.T. BAND LIVE

S.S.T.はライブ演奏を幾度かすでに経験しており，演奏や表情に余裕が感じられる。バランスのとれた音の「はめかた」は，もはやプロそのもの。以前よりギターが前面に押し出された感じで，とてもヘビーで迫力のある音になった。演奏曲目のほうはベストヒットの「AFTER BURNER」や新作「G-LOC」，またテーブル筐体の「ゴールデンアックス」，メガドライブの「ファンタシースターIII」とバラエティに富み，これらをフルアレンジで聴かせてくれる。

―― ZUNTATAは原曲重視，S.S.T.はアレンジ重視というのが私の印象だ。内容的には引き分けかな。次回こそは私も行きたい。　(善)

○サイトロンビデオコンサート

11月24日，東京は永田町の星陵会館にて，先日行われたS.S.T.とZUNTATAジョイントライブのビデオコンサートとファンの集いが開かれました。当日は多くのファンがつめかけ，上映が始まるやいなやみんな手拍子。「AFTER BURNER」ではライブそのままに立ち上がって拳をふりかざすモノまで現れるテンションの高さ。終了後はS.S.T.とZUNTATAのメンバーが現れてインタビュー。今後はもっとライブにも力を入れたいとのこと。ゲームミュージックも熱心なファンが増え，今後バンド活動も盛んになるはずです。ただ，コメントの最中にまで叫んでくる女性ファンがいて，メンバーも苦笑い。有名になるって大変だなあ。(浦)

多関節キャラの動きに注目だあ

Yamada Junji
山田 純二

ウルフの新作はサイドビュータイプのシューティングゲーム。オープニングはアニメーション，ゲームは多関節キャラの多彩な攻撃と，目にもうれしい作品だ。MIDIにも対応しているぞ。

最近，やけにシューティングゲームが多いなあ。先月号でも移植もの，オリジナル合わせて3本のゲームが紹介されていたし，さらには，「パロディウスだ！」までもが登場，なーんでまたこんなに集中しちゃってるんだろ。

で，ウルフ・チームからも完全オリジナル，バリバリのシューティングゲームが登場ってワケだ。これだけ数が揃っている（しかも一時期に集中している）ジャンルに挑んでいるのだから，この「ソル・フィース」に対するウルフ・チームの自信のほどがうかがえる。

さて，と。それでは，その自信がどこからくるのか，そして基本ともいえるサイドビュータイプのシューティングゲームを，ウルフがどうやってデザインしたか，じっくり見ていくことにしよう。

フルアニメーションだよん ◆◆◆◆◆◆

まずは，オープニングにおけるアニメーション。これは，全部で60秒ぐらいの全画面描き替えによるアニメーションだ。ふむ，X68000で見るアニメはなかなかの見ごたえだ。256×256モードの粗い画面ではあるが，そのぶん動きで見せてくれるので気持ちがいい。ラストの画面で思わずサ○ラ○ズのテロップを探してしまったほどだ。ゲームの本質とはまったく関係ない部分でこんなことをやっても不毛だ，と思う人もいるだろうが，個人的にはここまでこだわったのには好感が持てる。と，ここまでほめていながらいうのもなんだが，グラフィックはもう少し頑張ってほしかったな。

ちなみに，宣伝文句のフルアニメーションというのは，ディズニー映画などにおける1秒24コマの細かい動きという意味ではなく，オープニング全体を通してアニメーションしているといった意味のようである。

ここに注目! ◆◆◆◆◆◆◆◆◆◆◆◆◆◆

いきなり，趣味に走ってオープニングをほめてしまったが，もちろんゲーム自体も頑張っている。ゲーム構成は，さっきもいったようにサイドビュータイプのシューティングで，オプションによるパワーアップ，各面の最後にいるボスキャラを倒すと面クリアという，きわめてオーソドックスなタイプだ。と，これだけでは，いままで登場してきたシューティングゲームとなんら変わりはない。では，どこにこのゲームのウリがあるのか？

それは，スプライトの回転ルーチンを利用した多関節キャラなのである。各面ごとのボスや中ボスなどに，そのルーチンを利用していて，実に多彩な攻撃をしかけてくるのだ。たとえば1面では，メカニカルなカニのお化けが登場する。手や足の関節がわしゃわしゃと動き回りながら攻撃してくるのだが，この動きがとても自然で違和感がない。ほかのキャラクターでも効果的に使われていてずいぶん感心した。なるほど，これならウリにしたくなる。

オプション! ◆◆◆◆◆◆◆◆◆◆◆◆◆◆

ソル・フィースでは，4種類のパワーアップユニットがある。それぞれ，自機の上下に装備でき（取ったときの状況による），装備されたユニットは，水平位置から斜め40度近くまで，攻撃方向を変化させられるのが面白いところだ。ショットボタンを離しながら前進するとユニットは上下に広がり，後退するとユニットは閉じていく。ショットボタンが押されるとユニットはその位置で固定される。ここにも，スプライトの回転ルーチンを使っているので，微妙な角度で調節ができる。状況に応じて火力を前面に集中させたり分散させたりできるのは，なかなかいいアイデアだと思った。オプションのレーザーもきちんと斜めに飛んでくれて見ているだけで感心できる。

あと，このテのゲームにありがちなスピードアップのアイテムは，存在していない。自機のスピード設定はゲーム開始時のメニューにあるコンフィギュレーションモードで自分好みに選べるのである。これは，結構良心的。もっとも最高速にしてなければクリアは難しいかもしれないが……。欲をいえば，イメージファイトのようにゲーム中にも自分でスピードをコントロールできればもっとよかったんだけど。

バランスはどうなっている ◆◆◆◆◆◆

このゲーム，ノーコンティニューで全面クリアは難しいが，根性があれば普通の人でも大丈夫。難易度は少し高めだ。でも案ずることはない，敵の攻撃はそれなりに激しいが，一度くらい死んでも必ず復活チャンスがある。それに，一度死んだとしても，そのあとすぐブラスターの入ったパワーユニットが出てくるので，最低限のパワーアップはできるのだ。うん，良心的。

X68000用　5"2HD版3枚組　8,800円(税別)
ウルフ・チーム　☎03(5273)4795

一見強そうだが，実は弱いんだな，これが

各面の長さも適当だし，ボスキャラ到達までに適度に中ボスが現れたりして，プレイ中にだれてしまうこともない。それに，アイテムも派手さこそないが，最低限の装備でクリアできる。僕がこのゲームをプレイして，いちばん評価しているのはこの部分だ。「一度死んだらそれまでよ」的な僕はゲームは嫌いなのだ（ちなみに僕の好きなゲームは，「ドラゴンスピリット」と「オーダイン」。両方とも努力と根性さえあればノーマルショットでもクリアできるもん）。

▲撃墜後ミサイルを落とすヤなヤツ
▶もうなにもいわん，気合いでよけろ！

さて，ボスキャラ。これもよくできている。動きも凝っているし，攻撃パターンも多彩だ。また，ボスのつもりで根性入れて倒したキャラクターが実は中ボス!? と，びっくりすることもある。逆に，シューティングに慣れていない人には，多大なストレスを与えてしまうかもしれないけど。

そうそう，自機と地形との当たり判定は致命的ではなく，当たってしまうとただ単に跳ね返されるだけである。とはいえ，当たったときかなり予想外な方向に飛ばされてしまうので注意したほうがいいぞ。

面構成 ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

それでは，どのようにしてこのゲームが進行していくか見ていくことにしよう。

まず，ゲームはソル・フィースがワープアウトしてきた宇宙空間から開始される。最初の面とあって難しいところはない。じっくりプレイしよう。ボスはクラブ・ブレイン，はっきりいって弱い。攻撃を軽いフットワークでかわし，倒してしまおう。

先端部をさけて，進んでいこう

強制後方スクロールもあってなかなか楽しい

2面は，敵の大型兵器工場の中を進んでいく。ここでは，ふんだんに多関節キャラが登場する。地面をうにうにと進んでくるしゃくとり虫のようなヤツ，腕をぶんぶん振り回すラジオ体操野郎，そしてメカニカルな触手。この触手は，当たり判定が先端部分だけ，というのを覚えておこう。ボスについては，ヒ・ミ・ツ。

兵器工場をあとにして，3面は敵が打ち上げた人工太陽の近くを通過していくことになる。背景は美しいが，敵の攻撃はかなり激しい。特にミサイルを乱射しながら迫りくる2機の中ボスがきつい。この面のボスキャラの直前にちょっとした演出があるが，一見するとなにが起こったかわからない。まあ，気にするほどのこともないが。

4面は，突如現れた巨大戦艦との対決である。少し，ありふれた感じもするがなかなかの迫力。敵の砲台が撃ってくるレーザーをよけるのがちょっと大変かな。

5面では，冥王星付近にある建設中の敵の基地を進んでいく。途中，後ろからやっかいな置き土産野郎が出現。うまく誘導して障害物に当てるか，バックファイアーを使う，できれば後ろに回って倒そう。ボスの場面はかなり集中力が必要。

そして，終盤戦の6面。途中はともかく，この面のボスキャラは凄い。なんといっても「さては南京たますだれ」なのである。

え，なんのことかわからないって？ それはねぇ，敵のレーザー攻撃を見りゃわかるよ。で，レーザー攻撃をよけるだけでも苦しいのに，こいつの攻撃パターンがまたうんとこさあるのだ。ここまでくるとかなりしつこい。もう，攻撃のしがいがあるなんてもんじゃない。それだけにようやく倒したときの感動は忘れられないけど。

いよいよ最終面の7面では，地球が舞台で最後のボスとの対決……，は自分の目で確かめよう。

全体の印象は？ ◆◆◆◆◆◆◆◆◆◆◆

最終的にこのゲームの評価として，合格点をあげられると思う。場面ごとのつながりもなかなかいいし，ボスキャラもいい味だしていて，プレイヤーを驚かそうという努力が見られる。

しかし，画面全体の印象はあまりよくないと思った。きちんと2重スクロールもするし，一部ではラスタースクロールまでもやっている。でも，やっぱりパソコンゲームだなあ，と感じてしまうのである。そう感じさせてしまうのはやっぱり描き込みが足りないせいだろう。オープニングのグラフィックと同じようにね。ゲーム全体のバランスも取れているほうだと思うし内容もいい。あとはグラフィックデザイン。そこさえ直してしまえば，かなりアーケードゲームのレベルまで迫ることができるだろう。期待大ですね。

やっぱりMIDIだなあ

例によって例のごとく，このゲームもMIDI（MT-32系）の音源に対応している。一応，音色はオリジナルの音色を使っているようである。ウルフ・チームが制作しているゲームのFM音源を使用している曲は，以前より同じ音色を使っているのは皆さんご承知のとおり。そのため，別のゲームでもこれといってほかのゲームと目立って変わるようなことはない。ソル・フィースでもその事実は変わらないのである（いろいろと事情があるようだけれども，どうにかしてほしいものだね）。まあ，MIDIもっている人は1度聞いてみることを勧める。僕自身，試しに聞いてみたが音源が変わるとずいぶんと雰囲気が変わるものだと思った。と，いうことで曲作りのほうもさらに頑張ってくれるといいですね。

銀河の歴史がまた1ページ

Kaneko Syunichi
金子 俊一

すでに発売されている「銀河英雄伝説」の続編。前作と比べてシステムもパワーアップされ，ありがたいことに原作を知らない人のための指令官データやオート機能もついている。もちろんMIDI対応だ。

最近のシミュレーションゲームは力が入りすぎだ。デコレーションの部分ばかりがクローズアップされるようになり，マップの拡大化，ユニット数の増加，派手なビジュアルシーンといった具合に肥大化されてしまった。そう，シミュレーションは時間がかかって当たり前，24時間戦えますか的なゲームばかりになってしまったのだ。

それではいただけない。私はめんどくさがり屋なのである。中ダレしてしまうようなゲームではすぐに飽きてしまう。別にシミュレーションが嫌いなわけではない。その昔「サウザンティス号の冒険」というOh! PCに載っていたゲームがやりたくて，H. K. 氏の家に何泊もした私である。寝ている彼を横目に必死に舟を動かしていたものだ。

こんなのあってもいいんじゃない？◆◆

大袈裟なものじゃなくていい，手頃なシミュレーションウォーゲームが欲しいのだ。ほんの数時間で終わるシナリオ，適度に小さいマップ，限られたユニット数。これらの条件を満たせばコンピュータの思考時間が大幅に減る。なによりも大切にしたいのは無駄な時間の消費がないことだろう。はっきりいって待たされるのは，やだ。

で，銀河英雄伝説である。アニメーションを使った戦闘画面なのに1～3時間もあれば終わるシナリオ，現状のシミュレーションウォーゲームとしては，まさに短期決戦といえる。今回レビューしているのはその続編・銀河英雄伝説IIである。前作に比べマップは4倍，グラフィックも強化と，まるで肥大化への道を歩んでいるかのように見える。確かにプレイ時間は長くなっているが（1シナリオ2～3時間），まだバランスを崩しているレベルまでいっていない。救われた。

原作を知らない人のために◆◆◆

銀英伝シリーズは田中芳樹の同名小説が原作となっている。小説について私は外伝しか読んだことはないが，三国志に近いものを感じた。吉川英治の三国志にハマったことがある人ならば十分ハマれるだろう。同名のアニメもあり，グラフィック＆音楽関係はこちらのほうがマスターになっているようだ。ぜひご覧あれ。

銀英伝および銀英伝IIをプレイした感じでは，原作に対する知識は必要である。シナリオや登場人物の設定などは原作に忠実に作ってある。宇宙を舞台に銀河帝国と自由惑星同盟が戦っていることや，せめて登場人物の主だった面々がわかるぐらいでないと面白さは半減してしまう。目の前に黒色槍騎兵と呼ばれる艦隊が出現してもビビらないのは失礼にあたるというもんだ。まあ，ゲーム中にコマンドひとつで登場人物の経歴が見れるので，知らない人はこれを熟読しておくべし。ちなみに私は銀英伝を遊んでからアニメを見て本を買った人だ。

ウォーゲーム・銀英伝 ◆◆◆◆◆◆

一般的なシミュレーションウォーゲームと銀英伝IIとの違いを探ってみよう。ここでいう『一般的な』というのは光栄シリーズや大戦略シリーズのことである。

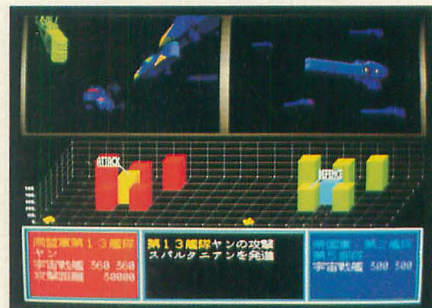
HEX戦ではない。現在地などは座標で表される。戦車とは違って同じ座標にいくつでも戦艦を置くことができる。だって宇宙だもん。Z軸だって無限である。

ターン方式が採用されているのは同じなのだが，**ターン終了と宣言するまでは何度でもやり直しがきく**。航路の変更をしたくなったら，そのまま新たに入力すればよい。最後に入力されたデータが選択される。X68000を買うときに，ACE-HDにしようかACEにしようか悩んでるうちにSUPERが出てしまったような優柔不断な人間にはお勧めのやり方である。

最大で32ユニットまで同時に扱える。しかし，毎ターンで32回も命令を出していたのでは時間と体力の浪費というもの。なぜなら**命令は全部のユニットに出さなくてもよい**からだ。このゲームには提督というものが存在している。1人の提督には大抵4～5ユニットが指揮下にあり，いっしょに行動している。これが1艦隊の単位である。つまり，提督ユニットに指示を出せば5～6ユニット同時に命令を出せるのだ。

でもって，**命令は毎ターン出さなくてもよい**。ユニットを移動させたかったら，そのユニットをクリック。メニューから移動コマンドを選んで，持っていきたい地点をクリックするだけだ。その移動には何ターン必要か教えてくれる。あとはその移動が終わるまで放っておけばいい。

さらに，**提督に作戦を伝えれば，作戦終了まで勝手にやってくれる**。たとえば「ここと，ここを通ってあの惑星を占領してくるように。あとはよきにはからえ」などと命令すると，そのとおりに実行してくれる。もちろん，途中での作戦変更や中止もできる。極端な話，1ターン目に作戦とそのコ

X68000用 5"2HD版 4枚組 9,800円（税別）
ボーステック　☎03（708）4711

立体棒グラフで艦隊の陣形をつかめ！

ースだけ決めておけばあとは放っといてもカタがついてるはずだ。うん，らくちん。

陣取りゲームなのね ◆◆◆◆◆◆◆

この世の中には帝国軍と同盟軍しか存在しない。どちらかを選び，勝利条件に達したほうが勝ちだ。非常に簡単なルールである。基本的に勝利条件とはマップ上にあるすべての惑星を占領すること，もしくは敵が全面撤退するまで叩き潰すことだ。そういった意味では，このゲームは見事に陣取りゲームである。しかし，ただの陣取りゲームでないことはシナリオ制が導入されていることからも想像がつく。シナリオごとに異なった目的があり，これも勝利条件になるのだ。

帝国軍を選んだ場合，特定の惑星の占領が主な目的となり，同盟軍だと，その惑星を防衛することが勝利条件になっている。つまり，完全制圧ではなくても軍事的に重要な惑星だけを占領下とすれば，そのシナリオは勝利したことと認めてくれるわけだ。無駄な時間の浪費につながらない方法である。シナリオは5本，それぞれに帝国側，同盟側の勝利条件が決まっている。

で，これらを通しでやるのがキャンペーンモード。キャンペーンモードでは特別ルールがあって，これまた遊び心をくすぐるものがある。そのルールとは，提督にレベルがあって，戦闘結果によってレベルが上がるのである。レベルが上がると先制攻撃をかけやすくなる，旗艦部隊の艦数が増えるという恩恵にあずかれるというもの。

彼がラインハルト。帝国軍の英雄である

自由惑星同盟の父、ハイネセン。これはデモ

ゲームの途中で経歴がわかる

そのほかにも，「以前のシナリオで占領した惑星はその後のシナリオで登場する場合にちゃんと自軍のものになっている」とか，「１度戦死した提督はそのキャンペーンには出られない」などがシナリオ単体でやるのとは異なってくる。

さらに最終シナリオで勝利すればエンディングも用意されているらしい。まさにいたれりつくせりである。

おりこうファジーじゃないけれど ◆◆

嬉しいことにいろんなコマンドに自動という選択枝がついている。たとえば補給艦で補給コマンドを実行した場合，本来プレイヤーは補給が必要な艦隊をいちいち選ばなければいけないのだが，自動にしてしまえば勝手に選んで補給をしてくれる。極めつけは「ゲームそのもの全自動」というやつである。アニメーションモードにしておけば立派な環境ソフトになっているわけだ。初心者だったら全自動モードを見ながら研究すればいい。きっと役に立つはずだ。

コンピュータのアルゴリズムは秀逸とはいいがたいが，十分に楽しめる。予測する楽しみがあるというのだろうか。マップ自体はすべて表示されているのだが，敵のユニットは索敵をして見つけた艦のみが表示されるのである。気づいたときには囲まれていたなんてことも起こりうるのだ。

敵艦隊のコースがわからないまま目的の惑星に前進するのはなかなかに怖いものがある。たまたま遭遇した敵が不敗の魔術師ヤン・ウェンリーの艦隊だったら……。攻め込んでくる提督が常勝の天才ラインハルト・フォン・ミューゼルだったら……。考えただけでも鳥肌が立つ。このゲームには見えない恐怖が常にある。その１歩先は闇という点がこのシステムの最大の面白さにつながっているのだ。

ちなみにコンピュータのアルゴリズムに飽きた人向けではないのだろうが，銀英伝IIになって対戦モードも用意された。ポピュラスが流行ったあとなので即席的な気も

ほしい情報が入る。親切設計はうれしい

するが，X68000を２台も必要としないからよしとしよう。

音楽性は??? ◆◆◆◆◆◆◆◆◆

最後に音楽について触れておこう。内蔵音源のほかにMIDI（MT-32に対応）がある。HELPキーを押しながら立ち上げればMIDIボードがあってもオープニングからFM音源が鳴る。もちろん，ゲーム中ならコマンドの選択でFM音源かMIDIかを選択できる。MIDIボードがささっているとFM音源は鳴らなかった前作と比べてもわかるように，システムとしては着実に進歩しているし，ユーザーの声もずいぶん反映されているようだ。これはとってもよいことだと思う。

しかし，肝心の音楽に関してはちょっといただけない。おそらくアニメの銀河英雄伝説のサントラ版か何かからの選曲なのだろうが，耳慣れしてないこともあってイマイチである。クラシックを使っていた前作のほうが聴きごたえがあった。効果音においては合格点なので，BGMを選択しないで自分の好きな音楽をかけるのもいいだろう。お勧めはベートーベンの交響曲３番・英雄である。前作に使われていた曲なので違和感がまったくないのがよい。

趣味の総評

フルマウスオペレーションを可能としていて，なかなかに気持ちがいい。艦隊をマウスひとつで運用してしまうあたりは，まさに「気分は艦隊指令官」といった感じである。そのマウスのシステムも，以前に増して磨きがかかってきている。左クリックが決定及び確認，右クリックが取り消しという姿勢がすべてにおいて貫き通してある。まるで重箱の隅をつつっきまわしたようなソフトで，いたるところに気配りが見られる。うん，買いだね。

総合評価	0　　　5　　　10
システム	★★★★★★★★
グラフィック	★★★★★★★
サウンド	★★★★
インタフェイス	★★★★★★★★
ラインハルト万歳	★★★★★★★★★★

皇帝陛下はマウスを司るのである

Komura Satoshi
古村 聡

このシミュレーションゲームがX68000に移植されると聞いて，忠犬ハチ公のように待っていた人もいることでしょう。いろいろな機種を経て，やっとX68000に，ということで完成度にも期待できそうですね。

X68000にシュヴァルツシルトが移植されました。このゲームはプレイヤーがサンクリ星国皇帝となり，工場の建設，宇宙戦艦の建造などを行いつつ，サンクリ星国を安全で強大なものにしていき，最終的にはジロ星団宇宙全体を平和にするのが目的，というシミュレーションゲームです。先にPC-9801，PC-8801，MSX版が出ていて，しかもなかなかにうまいシナリオ運びで評判がよかったので，ご存じの方も多いことでしょう。X68000用もずいぶん前から発売が決定していたのですが，完成までには結構時間がかかったようです。

で，レビューをやるためにこのゲームを始めたのですが，実をいうと原稿を書く時間が惜しいほど，ハマってしまっているんですよ。仕事だというのを忘れて，シャカリキになって遊びまくっていたりするんです。

吾輩は陛下である ◆◆◆◆◆◆◆◆◆◆

吾輩が大銀河シュヴァルツシルト・ジロ星団，サンクリ星国の皇帝である。皇帝陛下様と呼べ！　ううん，気持ちがよい。ふふふ。私はこれから楽して暮らすのである。なんてったって皇帝陛下様なのである。

む，これ，そこで何を難しい顔をしておる？　あ，なになに，皇帝なんだから仕事が山のようにあるのか。うむ，わかったわかった。よきにはからえ。

えっと，まず何ができるのだ？　ふむふむ。画面右上のメニューにできることが書いてあるわけだな。うむうむ，艦隊の移動，演習。いま持っている宇宙戦艦隊に命令を下せるわけだな。あれ？　艦隊は4つしか持てないのか？　正規の艦隊が4つと予備の無所属艦隊が2つで合わせて6つまでなのか。ふむ，それでは，どうやって戦力アップを図るのだ？

ほほう。戦力アップを狙うためには2つの方向があるとな。ひとつは戦艦の数を増やすこと，そして，もうひとつは戦艦のモデルチェンジ，すなわち性能を上げること（当然，性能が上がるのは新しいモデルになってから建造された艦のみ）なわけだな。ただし，違う型の戦艦は同じ艦隊には所属できないのか。まあ，型が新しい艦を大量に配備すればいいわけだがな。そうすると金が足りなくなる？　難しいところだのう。

ところで，平和的な皇帝のために，戦争をしなくてすむ方法というのはないのか。えっと，メニューを見てと。おお，外交なんてものがあるではないか。クリック。

おや，「反乱勃発による政情不安のため，諸外国との外交活動はまったく不可能です」だと。するとまずなにがなんでも反乱を武力鎮圧しなくてはいかんわけだな。よし，しかたがない。反乱軍をクリックして情報を見てみると。うっきゃーっ，反乱軍は2つも先のモデルの戦艦ではないか。おまけに反乱軍はどっかの星からあやしげな援助を受けてるという話じゃないか。こんなんで勝てるんかいな，まったく。

まあ，とりあえず戦艦のモデルチェンジをして，同時にいまのモデルの戦艦も増産してと。

よし，武器も揃ったところで，反乱軍鎮圧に出撃！

ふたを開ければ ◆◆◆◆◆◆◆◆◆◆◆

吾輩が大銀河シュヴァルツシルト統一を成し遂げた偉大な皇帝なのである。皇帝陛下様と呼べ！　うーん，何回やっても気持ちがよいぞ。

いまや大銀河シュヴァルツシルトを統一してしまったのである。統一というとちょっと語弊があるかもしれないな。とりあえずもう反乱を起こすものがいなくなった，というのが正解かもしれないのだが，とにかく戦いに勝ったのである。いやいや天下太平めでたいめでたい。

実際やってみると，楽勝であった。戦闘モードでの戦いの勝敗は自分の星に攻撃をされてしまうと負け，敵の旗艦を倒してしまえばこっちの勝ち，といういたって簡単なルールなのであった。だから，よっぽど敵のほうが戦力的に強すぎるなんてとき以外は，敵の惑星なり旗艦（印がついているのですぐわかる）を4艦隊で一気に攻め込むという戦法でなんとかなるのだ。

敵が攻めてきたときには攻守同盟を結ん

X68000用　5"2HD版2枚組　12,800円(税別)
工画堂スタジオ　☎03(353)7724

おっ，戦艦ができたのだな

情報はウィンドウ上に表示される

ている国の軍隊を呼ぶ，というのもいい戦法だ。なにしろ同盟国軍の艦隊も，全艦隊がとにかく敵の旗艦に突っ込んでいくので，自分の艦隊は惑星の近くに待機するようにしておけば，そこへ敵艦が来るころには敵旗艦は相当消耗している。そこで敵旗艦を叩けば，結構楽に敵を倒すことができるのである。ちなみにこの戦法はオペレーションが面倒臭くなったときなどにもなかなか有効である。同盟国が勝手に敵をやっつけてくれるから。

どちらにしても戦闘のときはそんなに深く悩む必要はない。このゲームではとにかく，「数の勝利」という言葉がぴったりくるのである。

反乱軍を鎮圧してしまうと，やっと他国との外交ができるようになる。同盟と不可侵条約に関しては積極的に行うようにすべきであろう。なにしろ，同盟軍は多ければ多いほど戦いの際にも有利だし，不可侵条約を結ぶことができるのなら，一度に相手にする国が減るわけだから有利になる。また，宣戦布告はそれほど自分から行う必要はない。すべての国がというわけではないが，ほとんどの場合，敵となる国は自分から宣戦布告を行ってくるからなのである。

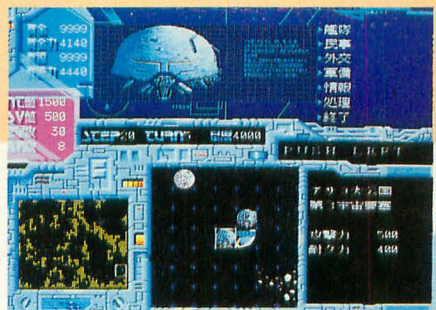
宣戦布告だとぉ，返り討ちにしてくれる

一難去って，また一難，新たな敵が……

宇宙要塞というのもアリ

とにかく，このゲームでは戦闘そのものではなく，投資の振り分け方でだいたい勝敗が決まってくる。戦艦をどれだけ持っているかで戦闘の結果の半分は決定されるといっていい。とりあえずは，いまある型の戦艦を増やし，いち段落したら新型の戦艦開発に投資して，それを徐々に配備していく。これが銀河皇帝への道である。

時間がかかっただけのことは……◆◆◆

さて，皇帝になりきるのはこれぐらいにしてと。感想なのでありますが，本当に面白いです。ハマります。まず，反乱軍と戦い終わると新たな敵が，そいつに勝つとさらに新たなる敵が……，という具合に進んでいくので，なかなかやっていて楽しいです。よくゲーム進行を考えて作られていると思います。そうそう，目の前の敵を倒すと次なる新たな敵が出てくる（当然，前の敵よりも強い）わけですから，目の前の敵に勝てそうだからといって安心してはいけないのですよね。ちょっと多め，新しめに戦艦を作っておくと，あとの展開がとても楽，かつスピーディになります。これが銀河統一のコツです。

戦闘場面はアニメーション，撃て撃てぇ！

あ，統一で思い出した。このシュヴァルツシルトなのですが，実をいうとシュヴァルツシルト銀河に敵なし！　……というさっきのところでゲームが終わりではないのです。というのも，実は○○で○○な敵が攻めてきて……（あんまりいっちゃうとまずいだろうな），とにかくここから先が最大のヤマ場であり，真のシュヴァルツシルトでもあるのです。

このゲーム，システム的にも大変よくできています。操作はフルマウスオペレーションでスイスイいけちゃうし，コマンドも比較的種類が少ないのでシミュレーションゲーム特有のかったるさはまったくありません。

また，グラフィック周りもX68000用に描き直されていてきれいになっています。外交や宣戦布告されたときには相手の国の元首の顔が出てきたり，戦艦ができたときなどには戦艦の絵が出てきます。戦闘時には，たとえば惑星を攻めるときには惑星にバシュバシュバシュッとミサイルを打ち込む場面が，しっかりとアニメーションしてくれます。このアニメーションのときにディスクを読みにいくんですが，結構読むのが速くて気にならない程度になっています。X68000の性能を生かした移植となっていて，システム的にも大変よくできている，と私は思います，はい。

一般ピープルおすすめゲームだ

シミュレーションゲームというと面倒臭い，ゲームの目標がわかりにくい，ゲームが単調になりやすい，といったことが起こりがちです。これはいままでシミュレーションゲームではある程度避けられない問題でした。シミュレーションゲームの特徴が“シミュレーション性＝現実の模倣”にある以上，現実の世界にある繁雑な事柄を引きずってしまうのはどうしてもやむをえないことですからね。

このシュヴァルツシルトはどこかの星々という虚構の世界のシミュレーションです。つまり，人が作り上げたフィクションの世界の話であって，現実をシミュレートしたものではありません。このゲームではそれがとてもよい方向に作用しています。ゲームの先が読みやすくなり，ひとつの目標が達成されたら，また次の……とうまくプレイヤーをゲームに引き込んでいくことに成功しています。

シミュレーションマニアの人であればゲームを行うこと自体をゲームの目的と捉えることもできるのでしょうが，普通の人の場合はそうもいかず，ゲームを進めるのになにか目標がないとやっていけないという人が多いと思います。

そんなわけでこのゲームはシミュレーションマニア以外の人にも絶対的にお勧めできる1本です。特にシミュレーションは面倒臭いと避けていた人にはシミュレーションゲームへの入り口として，ぜひともやっていただきたいと思います。

総合評価	0　　　5　　　10
とっつきやすさ	★★★★★★★★★
皇帝気分	★★★★★★★
おまたせ度	★★★★★★★★
オススメ度	★★★★★★★

クリスマスを暗いダンジョンで祝おう

――七面鳥の代わりに丸虫の肉で――

Ogikubo Kei
荻窪 圭

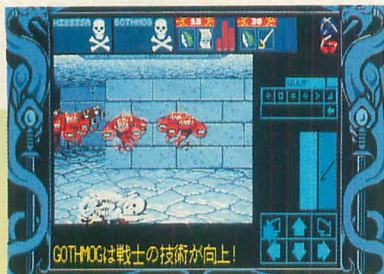
いよいよあのダンジョン・マスターの続編が一段と難しくなって登場。"続"というだけあって，もちろん前作のキャラクターデータを使うこともできる。フルアニメーションのオープニングも見ものだ。

caution !

前作のダンジョン・マスターを遊んでない方。本作品は前作を知らなくてもまったく問題なく遊べます。本当に大丈夫です。しかし，シューティングゲームを知らない人がいきなりナイアスやイメージファイトを始めてしまうようなものです。魔法を唱えるのにあたふたし，戦いにとまどい，ダンジョンの要所を見逃し，無数の傷を負うでしょう。ダンジョン・マスターを終わらせる必要はまったくありませんが，せめて，地下10階くらいまでは経験しているとずいぶん違います。ロード・カオスを無事融合させた人でも，私のように半年もブランクがあるとなかなか勘が取り戻せないので，しばらくダンジョン・マスターのいちばん成長したキャラクターディスクで遊んでみるのもいいでしょう。その際，ちいとばかりレベルをあげておくとよりGoodです。

*　*　*

本当はいやだったのだ。もう2度とあんな目はこりごりだったのだ。食料さえおぼつかない，ちょっと気を抜けば暗闇が襲い，曲がり角には怪物が息を潜め，傷を負い，いつ果てるとも知れぬ命といつ果てるとも知れぬ迷宮の持久走。

しかし，穴があれば入るのが男。古今東西そういうものなのである。穴があったら入れたい，じゃなかった，入りたいのだ。たとえそこがどんな怪物が潜むやもしれぬダンジョンであっても。

X68000用　5"2HD版3枚組　9,800円(税別)
ビクター音楽産業　☎03(423)7901

旅の道連れを決めること ◆◆◆◆◆◆

カオスの逆襲。カオスってのは混沌のこと。カオスが悪者ってのはエントロピーの法則を信じている私には少々気に入らないが，エントロピーが増大すること自体が悪なのかもしれない。私の部屋がすぐ汚くなってほこりが溜まるのはエントロピーの法則が働いたからにほかならないのだが，女の子はそれを見て"悪"だという。そんなものさ。私が悪いわけではないのだ。

逆襲してくるならこちらも返り討ちにしてくれよう。というわけで，旅の道連れを決める。どうするかは私の勝手。

選択肢は2つだ。新たに閉じ込められた強者（今度は勇者ではなく，チャンピオンだ）を4人選ぶか，前回の冒険で道連れにした者たちの手を再び借りるか。

今までの常識からいうと，前作のキャラクターを持ってきたほうが楽に進めるはずだ。しかし，である。そうはいかないところがダンジョン・マスターだな。それとも，私があまりにも成長させないまま終わらせてしまったからかもしれない。とにかく，弱い弱い。まあ，普通の人は前作のキャラクターを"MASTER"くらいにはしてるだろうから，大丈夫だろう。チャンピオンたちはみなEXPERTくらいだ。その代わり，ヘルスやスタミナが大きい。どっちもどっちだ。今回の新キャラクターは人間でないやつらが多いので，人間パーティを組みたいときはなおさら前作のキャラが欲しいだろう。今回登場したケンタウロスを使いたいという気もするが，私は一応義理と人情で前作のヒッッッッサとゴスモグとウーフとダルーーーを使ってやるのであった。彼らもいやだろうが，旅は道連れ世は情けってもんだ。

ちいとばかし，前作とシステムが違って，道連れを4人選ぶ牢獄と旅するダンジョンが分かれている。牢獄で4人選んでセーブするか（今回はなかなかエグいやつらばかりのフリークスだ），前作のデータディスクを用意したら，"冒険の準備"なるものをせねばならないのだ。

何はともあれ準備から，ってわけ。どうせなら，冒険の前にお買い物もしたかったけど，させてくれなかった。今回は完全手ぶら状態でダンジョン突入だ。

暗闇で丸虫に囲まれること ◆◆◆◆◆

突入。暗闇。ここはどこ。何者かに襲われている。誰だ。足元にたいまつ。拾って灯す。鉄格子。振り向く。食べられるが味はOOP！　な丸虫のアーマードバージョン！　毒毒毒。

ほうほうのていというやつ。では後世の勇者たちにメッセージを残そう。いきなりアーマード丸虫のド真ン中に現れるので，ぬさもとりあえず逃げるべし。アーマード丸虫は3匹しかいない。ただし，鉄格子の前はアーマード丸虫出現フットスイッチになっているので，格子を開ける鍵を見つけるまでは決して鉄格子には近づかぬよう。

試練。とりあえず逃げ，明かりを灯し，アーマード丸虫をやっつけ，やっつけ，やっ……つけようと思ったのはいいが，戦士は丸腰，魔法使いは……魔法を思い出せない。前作の魔法をそのまま使えるのに，魔法をメモした紙をなくしてしまったのだ。ファイアーボールの魔法はどこだ。しかたがないのでダンジョン・マスターガイドブックに頼ってしまった。

レッドドラゴンにびびるの図

魔法を駆使し，3匹のアーマード丸虫を倒し，肉を懐へしまい，部屋にあるアイテムをかき集め，よく眠り，体調を整え，鉄格子OPEN。

ここからが冒険の始まり。

無名戦士の骨のこと ◆◆◆◆◆◆◆◆

前作ダンジョン・マスターのおかげで，マウス・トラックボールの左ボタンがいかれ，以来，両脇についているサブボタンを使っている。マウスを見ると，わしづかみにする癖がついてしまった。

昔冒険したダンジョンは下へ向かって進む猪突猛進だった。そのフロアを制覇したら下へ，下へ，深く，深く。

CHAOS STRIKES BACK（邦題：カオスの逆襲）はそんなあまい場所ではない。いきなり地下5階である。上へいけばいいのか下へいけばいいのかわからず，目に見えない落とし穴はあり，目に見えないワープゾーンはあり，いきなりレッドドラゴンはいる。ないのは愛と慈悲だけだ。

最初のトラップは空フラスコ。空フラスコが欲しい。ないと，怪我も直せない。取りにいくと，必ず穴に落ちねばならない。どうするか。答え。フラスコはきっとほかにもあるさ。

魔法の地図。これが諸悪の根源かもしれない。右手に持つと，7ブロック分の地図が見られる。これを方眼紙にでも写し取ればいいのだが，いかんせん，マッピング習慣のない悲しさ，つい地図に頼ってしまい，観察や目印設置がおろそかになる。

今回のダンジョンは恐怖だ。あまりにも先は長く，あまりにも複雑で，あまりにも苛酷。マッピングなしでどこまで行くかに挑戦。ひとつのフロアがいくつにも分割され，そのそれぞれに階段があり，壁には壁のようで壁でない壁がたくさんあるのだ。

地図巻紙の上に，呪文をのせる。壁でない壁や見えない落とし穴（目を凝らしていれば魔法を使わなくてもわかるが）が見える。ダンジョンにいるモンスターの場所がわかる。そういう呪文がオリジナルのAtari-ST版に加えられている。本家本元に手を加えるのは望むところではないのだが，許そう。それほどカオスの逆襲は厄介。

歩いていると，ひとマスの部屋に出た。ワープ装置が設置されており，四方，選んだほうへワープする。どちらにも復活の祭壇があり，祭壇にアイテムがひとつずつ置いてある。どれを選ぶかでとりあえずの将来が決まる。

運の良し悪しはない。強いていえば，闇の剣は呪われていることくらいだ。この祭壇選択の場には再びみたびおとずれることになるだろう。

こいつが魔法の地図だぜい，の図

キャラクターエディタの図

ひとつを選んでさらに地下へと降りる。こん棒を持ったやつらを蹴ちらし，さらに下へ降りる。広い部屋だ。そろそろ地下8階くらいだろう。骨が落ちている。無名戦士の骨だ。思ったとおり，レッドドラゴンがいた。やつがのろまなのをいいことに，足元をすりぬける。レッドドラゴンと戦う元気はない。

壁でない壁を魔法で見つけ，通り抜けるとワープゾーン。そこへとびこむと，再び祭壇選択の場。

道は無数だがひとつもないこと ◆◆

はあ，はあ。み，みずぅ。はあ，はあ。く，くいものぉ。確か，食い物の元，スクリーマーの宝庫があったはずだ。あれはどこだった？　あ，思い出せない。み，みず。あ，水だ。前はなかったのに，いつの間に水が？　苔が浮いているけど文句はいえない。

あ，閉じ込められた。巻物がある。奇跡を祈れ。祈る。祈る。

明日はどこにある。

紹介するのこと ◆◆◆◆◆◆◆◆◆◆◆

というわけで，カオスの逆襲はいきなり3者択一メニューから始まる。ひとつがフルアニメーションのオープニング。続いて，冒険の準備。ここではダンジョンに突入させるキャラクターのディスクをロードして，冒険の準備をする。その際，名前や姿形をエディットすることができる。そういうことらしい。だから，魔法使いサリーとか僧侶の一休さんとか戦士のバトロイド・バルキリーとか忍者部隊月光なんてのも作れるわけだ。

3つめがGO TO ダンジョン。例によって入り口の扉が現れるわけだが，新しく始めるときは道連れの選択になる。道連れを選択したらセーブして終了して冒険の準備をしてもう一度ダンジョン突入をする必要があるのだな。

今回新しく付け加わったのは，ところどころのBGMと，新しい魔法と効果音の充実（ときのこえはちゃんとあげるし，壁を叩くと音がするようになった）だけ。それから24KHzディスプレイ対応になった。24KHz対応ディスプレイを使っている人は，24KHz（つまり，PC-9801やFM-TOWNSと同じような画面）でも遊べるようになったのも変更点だ。

あとは前作とまったく一緒。あ，BGMがついたと聞いてラッキーって思っただろ。ラッキーじゃないよまったく。ダンジョン・マスターはBGM不要のゲームなんだから余計なことはしないでもらいたい。無音の緊張感が楽しいのだ。

ダンジョン・マスターは逆襲してきた。広さより複雑さを重視したマップとともに。トラップの数とプレイヤー心理を逆なでするいやらしさは天下一品といえるだろう。

カオスの逆襲をひとことでいうと

かつて私は，ダンジョン・マスターを指してアメリカの冒険文化がゲームになったものと評した。それは今回も貫かれている。ただし，前作のような教育的配慮がまったくない。いきなりバカ強いやつと遭ったり，その先で弱い食用モンスターと遭ったりという感じだ。道は常に選択されるものであり，他人の意見は参考にならない。丸腰で始めるため，前作後半よりずっと軽い情けない装備でレッドドラゴンと出会ったりする。2Mバイトあるとオンメモリで動くことや，ディスク1枚であれだけ遊べるという最高の売りは健在だ。

総合評価	0　　5　　10
スリル	★★★★★★★★★
サバイバル	★★★★★★★★★★
教育的配慮	★★★
手軽さ	★★★★★★★★★
手軽に始めたあとの後悔	★★★★★★★★★

みんな知ってる定番野球ゲーム

Kageyama Hiroaki
影山 裕昭

簡単な操作法で野球の楽しみが味わえると，大人気の「ワールドスタジアム」。その“ワースタ”がついにX68000にも移植されました。これでストーブリーグも友達を集めて野球大会でもやれば，さみしくないかな？

今年の日本シリーズはあっという間に終わったようですね。ああ，つまらない。僕は巨人ファンですから。プロ野球シーズンも終わってしまいましたが，野球ゲームのシーズンはこれからです。X68000にもやっとワールドスタジアムが発売されました。ワールドスタジアム，通称“ワースタ”はファミコンゲームの不朽の名作“ファミリースタジアム”のお兄さんとでもいうべきゲームで，野球ゲームの決定版です。

いない選手は作れ ◆◆◆◆◆◆◆◆◆◆

ワースタには12球団が登録されていて，アーバンリーグ，カントリーリーグという呼び名になっています。球団名や選手名は実在のものをもじったもので，これがなかなか笑えてしまうのです(ガイアンツの“まきばり”でウケてしまったのは私だけ？)。1球団は投手が5人，レギュラーの野手が8人，そして代打が5人いるので合計18人です。どの球団にも代打になつかしの名選手がいて，それを使えばどのチームでも割と遜色なく戦うことができます。

ところで，バッカルーズ（近鉄のこと）を選んでも，投手陣の中に野茂の名前がありません。野茂のいないバッカルーズなんて彼女のいないイブの夜のようなもの。ちょっと寂しいですよね。よく選手を見てみると，なんとなくデータが古いようです。阪神にはあの恐ろしいバースがいるし。

でも，このワースタには選手データを変更できるエディットモードがあるからご安心召され。細かいところまで変更できるから，どんな選手も簡単に作れます。選手名，利き腕，バッティングフォーム，肌の色，バットの種類，ホームラン数，以上が野手，投手共通に変更できるもの。これに加えて，野手は打率，打撃力，足の速さ，投手はピッチングフォーム，防御率，スタミナ，球速，変化球の曲がり具合を変更できます。

これだけたくさん変更できると十分といえるでしょう。こだわる人なら選手年鑑やスポーツ新聞などを片手に，各球団のデータをそっくり打ち込むのも面白そうです。

簡単とはいったものの，数が多いから全選手のデータの変更にはそれなりの時間と労力を必要とします。その点，ちゃんとデータの保存ができるようになっているのはありがたいことです（当たり前？）。

球場は後楽園，甲子園，メジャーの中から自分の好きなものを選択できます。球場のグラフィックはどれもきれいで，後楽園や甲子園は本物をもじった広告などがあって雰囲気が出ています。メジャーは本当に広いし，イニングの合間には飛行船が飛んでいたりして，アメリカンな感じです。

さて，操作方法はといえば，ファミスタと同じです。しかし，ジョイパッドやジョイスティックによってはボタンがファミコンなどと逆になっているものがあります。これはファミコンなどでやり込んでいるものにはちょっとつらい。ソフトウェアでのボタンの逆転がほしいところです。

ルールは実際の野球と違うところがあります。ランナーのいない塁に牽制球を投げてもボークにはなりません。また，10点差がつくとコールドゲームになります。

X68000用　5"2HD 2枚組　8,800円(税別)
SPS　☎0245(45)5777

ひとりのときは ◆◆◆◆◆◆◆◆◆◆◆◆

スポーツゲームは友達と対戦して遊ぶのが真骨頂。それでも，ひとりで遊ばなくてはならないときもあるから，コンピュータの強さも気になるところです。そこでコンピュータの実力を分析してみました。

バッティングはまあまあうまい，ので文句なし。それよりも，守備が問題。フライを捕るのはうまいけど，ごくたまにライナー性の打球が飛ぶと，頭の上を抜かれてしまうことがあります。決して捕れない打球ではないのに。それとゴロの処理ははっきりいってヘタ。後ろに回り込みながら捕るということをしないので，守備範囲が狭い。コンピュータだから守備は完璧だと考えてしまうのは，まったくの誤解です。そのうえ，相変わらず（ファミスタのときから）捕球したあとの送球がまぬけです。

「ピッチャー，振りかぶって，投げました。カキーン。打った。これは大きい。右中間まっぷたつ。ランナーは1塁を回って2塁へ。センターがやっと追いついて，セカンドへ返球しました。おーっと，これを見た打者走者はすかさず3塁へ進塁，ゆうゆうセーフ」

この場面では，打者走者が2塁ベースを踏む直前にセンターが捕球しました。ここでプログラムは返球先を決めるのに，打者走者の位置を調べます。すると走者は2塁ベースを踏む直前，つまり1塁と2塁の間にいることを確認したので，2塁へ返球したのです。これが人間だったらどうでしょう。いまから2塁へ返球しても間に合わない。しょうがないから，3塁へ返球しておこう，となるはずだ。ランナーがある塁にかなり近かったら，ひとつ先の塁に返球する

各選手の設定。こんなにパラメータが多い

ようにプログラムすることも簡単なはずなのに。全体的にコンピュータ側は先を読んだプレイができないのです。ワースタは2人でワイワイとはしゃいで遊ぶのが面白いゲームだから，コンピュータの思考ルーチンはどうでもいいという感はありますが……，ね。

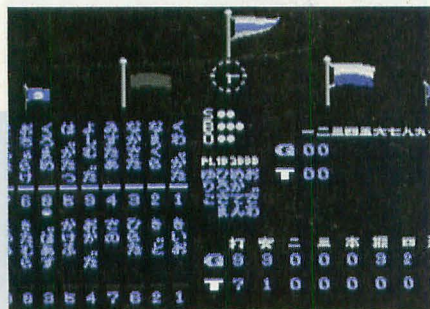
なんといっても対戦◆◆◆◆◆◆◆◆◆

というわけで，コンピュータと対戦して遊べるのは，操作方法に慣れていない最初のうちだけ。やっぱり友達を引っ張ってきて対戦するのが正しいワースタの遊び方でしょう。

人間が相手となると気合いの入れ方もがぜん変わってきます。やっぱり，やるからには勝ちたいじゃん。ここでちょいと伝統的な常識技を紹介しておきましょう。

まずは，速い送球。これはレバーを倒すのと同時にAボタンを押せばいい。きわどいタイミングの内野ゴロや，外野からバックホームするときに使うと有効な技です。

それから2塁へ盗塁した走者を刺すとき，普通なら上＋Aですが，下＋Aを押します。下＋Aはバックホームですが，捕手は投手からの捕球時はホームベースの後ろに立っているので，誰もいないホームに投げてしまうのです。するとあろうことか，投げた球はそのままゴロゴロと2塁ベースまで転がっていくわけ（球が高く浮かないから速い)。投手が捕ってしまわないようにしないといけませんが，この送球は使えます。

それとピッチングにもヒットを打たれないようなコツがあります。難しいことはなく，ストライクを投げるときはコーナーいっぱいを狙って投げる。ときたまボール球やフォークを投げるとさらに効果的です。これだけのことで，バットに当たってもなかなかヒットが出なくなります。

スコアボード，主審の名はゆりこ

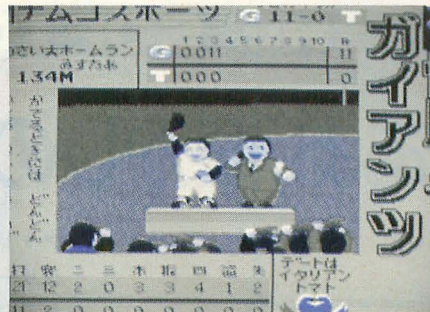
三振するとこっちを向いてくやしがる

ホームラン！　さすがは「おうさだ」

守備も最初はフライを捕るのが難しいけど，ボールの影の移動速度と，キュイィィィンという音から高度を推定して，すばやく予想落下地点へ選手を移動させます。捕れないと思ったら前に出るのを止め，すこし後ろに下がって待ち，落下して1バウンド以上してから捕るようにしましょう。バッティングはなんでもかんでも引っ張るよりは，流し打ちしたほうがヒットが出やすいようです。

と，これだけ頭に叩き込んだら，最初はコンピュータと対戦してみましょう。ファミスタをやったことのない人なら，おそらく苦戦することでしょう。

試合開始時には「プレイボール」の掛け声がかかる。BGMの流れる中，「ストライク」，「ボール」，「アウト」，「セーフ」とサンプリングされた女の子の声がスピーカーから聞こえてきます。審判は女の子のようです（スコアボード上の審判の名前でもわかる)。なんともナムコらしい発想。ほかにもデッドボールだと「いてっ」，「このやろー！」，ホームランを打つと「入った，入った，ホームラーン」という声が流れます。

そうそう，ホームランを打ったときの演出がとても気持ちいいのだ。ハデハデな音

試合の結果はナムコスポーツで

楽が流れて，画面にはデカデカとHOMERUNと表示され，それがネオンのように瞬いて，おまけに飛距離まで教えてくれる。逆転ホームランだったりすると，ついちょっと前まで肩に入っていた力も抜けて，ホッとして悦に入ってしまいますよね。

チェンジのときには，スコアボードが表示されます。時計もちゃんと動いているから試合時間（現実時間とは違うぞ）もわかる。

そして，試合が終了すると，ナムコ新聞が今日の試合結果を伝えてくれる。左側に書かれる3行の文章が読んでいておかしい。デートはイタリアントマト，という宣伝がなにか怪しい。

またまた要2Mバイト◆◆◆◆◆◆◆◆

うすうす気づいたと思いますが，このゲームもワールドコートに引き続き，要2Mバイトです。たぶんサンプリングデータがメモリを占有するのでしょう。サンプリングの数を減らしても（なくても）ワースタが遊びたいという人はたくさんいると思います。

SPSさんもワールドコートのときに同じような意見を聞いていたと思うのですが，1Mバイトでも遊べるようにはできなかったのでしょうか。2Mバイトあるときにのみ，サンプリング音などが流れるなどというふうにすればいいと思うのですが。遊べるゲームだけになんとかしてほしかったところです。

やっぱり，2人で

ワールドスタジアムは元祖のファミスタがファミコン用だったということもあって，非常に取っつきやすく，それでいて奥の深いゲームです。ゲームの中身はそういうことでよくできていますし，声が出る，球場のグラフィックもよくできている，動きもいいなどと，見栄えもよくなっています。観衆はちゃんと人間の形をしているし，ホームランを打てば手を上げてよろこんで（怒って）くれます。移植はほぼ完璧といっていいでしょう。ファミコンやPCエンジンでやっても楽しめるゲームなので，「わざわざX68000でやることもない」と思う人もいることでしょう。しかし，パソコン版の特徴としてエディットモードがあるので，また違った楽しみが味わえるのではないでしょうか。

ああ，懐かしのフェアリーランド

Takahashi Tetsushi
高橋 哲史

アクションRPGの元祖ともいうべき，あの“ハイドライド”シリーズがX68000に初登場です。1作目と2作目は移植されていないので，初めて耳にする人もいるかもしれませんが，知っている人は本当に懐かしいですよね。

ハイドライド3がついにX68000にも上陸です。いやあ，懐かしい。実はもう2年くらい前にやったんですよね，これ。その当時はX1Dユーザーだったのですが（このへんが時代を感じさせる），どのソフトも「X1turbo専用」になってきていて，愛機X1Dと泣き寝入りしていました。そこに「ノーマルX1でも動く」ということで，感激して飛びついたのをいまでも覚えています。

さて，知らない人も結構いるかもしれませんので，ちょこっと“ハイドライド”について，ご説明を。アクションRPG（以下，ARPGと略）の紹介などで，「このゲームはハイドライドタイプのARPGで……」と引き合いに出されるように，この“ハイドライド”シリーズはARPGの元祖といえるものです。出た当時は「ドラゴンスレイヤーか，ハイドライドか」ってなもんで，おおいに話題になりました。“ハイドライド”は，まさに多くのRPGのひとつの原型となっている名作中の名作ゲームなのです。

また，今回移植されたのはPC-9801用に出ていたハイドライド3SV（スペシャルバージョン）ということで，X1版をやった私にも楽しめそうです。

勇者でも最初は弱い ◆◆◆◆◆◆◆◆

さて，ハイドライドも3作目からはそれまでの伝統(?)が破られ，自分のキャラに名前がつけられます（2作目までは無条件で「ジム」だった)。職業は戦士，強盗，僧侶，修道士の4種類。温和で優しい性格の高橋くんは(?)「戦士」を選んで勇者「りゅう」君を作ります。キャラ作成画面にいっぱいのパラメータが表示されました。なになに，体力に腕力，魔力……。

高橋「まあ，こんなもんでいいか。面倒臭いし」

りゅう「あーあ，ここでじっくり選ばないと，あとで苦労するかも。なんか，困難な旅の始まりを感じるなあ。実際に戦うのは僕なんだからね」

そんな心配をよそに，まず街に入って装備を揃えます。当然のことながらあまり高いものは買えません。最初はみんな貧乏なのです。地道に「食人樹」「ジェリー」を倒して，お金と経験値を貯めていきましょう。

高橋「うおー，てめえら金よこせーっ！　バシバシッ，ブシュー」

りゅう「ああっ，見かけは似ているけど『樹木の精』さんとか『スライム』くんとかはいい人だから倒しちゃいけないのに！　勝手に体が動くーっ」

真の勇者は戦いの最中でも理性を失ってはいけません。ハイドライドには「善良なモンスター」と「悪いモンスター」がいるので注意しなくてはいけないのです。間違って，善良なモンスター（近づいても攻撃してこない）を倒してしまうとパラメータのうちの「精神力」がぐっと下がってしまいます。そうなると，お店の人なんかが品物をべらぼーに高く売りつけてくれるので，とても悲しい思いをしなくてはならないのです。

高橋「ふっ，勇者はいつも孤独なのさ」

りゅう「なんか違うんじゃない？」

さあ，ほどよくお金も貯まりました。思う存分にお買い物です。ええと，鎧に楯に槍に体力回復剤に……。あら，食料も安いわね，3つくらい包んでくださる？

高橋「あれ，お店から出たとたんに一歩も動けなくなったぞ。どうしたんだ？　そうか，魔王の陰謀だな。正々堂々と出てきて勝負しやがれ！　ビュンビュン」(と剣を振り回す)

りゅう「に，荷物が重いんだよーっっ」

ちぇーっくっ！　そうです，このゲームではアイテムの「重さ」が考慮されていて，あまり多くの荷物を持ちすぎるとその「重さ」のために動けなくなってしまうのです。この制限はある意味ではうっとうしいものですが，納得いく設定だし現実味が味わえていいと思います。

高橋「ちぇ，力のない奴だなー。しょうがない。食料捨てて，槍も捨てて……。ああ，もったいない」

りゅう「勝手なこと言ってくれるなあ。それなら最初のキャラクタ設定で，持てる重量というパラメータを大きくしてくれればよかったのに」

X68000用　5"2HD版2枚組　7,200円(税込)
ブラザー工業(TAKERU)☎052(824)2493

武器屋，強そうなのはやっぱり高い

暗闇せまるころ，ここは教会の前

ハーベルの塔に◆◆◆◆◆◆◆◆◆◆◆

ハーベルの塔というと東鳩の「ハーベスト」を思い出してしまうのは私だけでしょうか？　なにはともあれ，いよいよ広告にもでかでかと登場している「ハーベルの塔」に突入です。この塔は前半の要ともいえる重要なポイントですので心して挑みましょう。

高橋「200階もあるって，面倒臭いな。やだな」

りゅう「僕なんか実際に歩くんだよ。そっちは指を動かすだけなんだから，しっかりやる気出してよ」

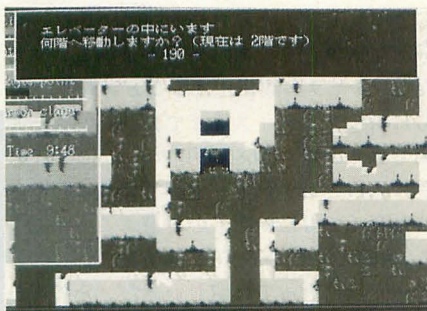
200階建てというのは間違いないのですがさすがに「全階巡らなきゃ解けません」ってなことにはなってないので安心してくださいね。肝心なのは○階のスイッチと宝箱と最上階の○の石ですから。あ，あとさすがに中が迷宮化してるだけあって抜け道とかなんかもあったりします。たまにそこからモンスター君がいきなり通り抜けて登場してくれるのでびびってしまいますが。

高橋「だあーっ，ボスだーっ！　このっこのっ，死ねー」

りゅう「うわーん，そんな無謀な戦い方しないでよ。痛いよー」

幾度となく非業の死を遂げながらも（誰のせいで？），冒険は徐々に進展していくのでありました。

天空の街◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

高橋「すごい，ラピュタは本当にあったんだっ！」

ハーベルの塔はエレベータつき

天空の城の王様はなにやらもの憂げ

地下の街，どうやってくるのかはヒミツ

洞窟の中はスポット処理で表示

りゅう「……」

天空の街にご到着です。え，どうやって来たかって？　それは自分で考えましょう。案外簡単に来られると思うんですけど。あ，当然ボスを倒さないと来られませんよ。ま，これ以上はノーコメント。

ここでは高度な呪文を修得できる「魔術師の館」があったり，薬草をただでくれる気前のいいお城があったりとなかなかいいところです。それと実は「水の城」へはここから行ったりするんですねー。これは注意しないとよくわからないと思いますが，注意していればすぐわかります（ミもフタもない言い方）。肝心なのは思い切りと勇気です。

高橋「わかった。こうすればいいんだ。えいっ！」

りゅう「きゃあああっっ」

りゅう君がさまざまな試練を乗り越えてくれるおかげで我々に新たな明日がひらけるのです。ありがとう！　勇者りゅう君！

りゅう「もう身も心もボロボロ……」

水の城……と見せかけて地下の街◆◆

話の流れに大幅に逆らって，なぜか今度は地下の街です。

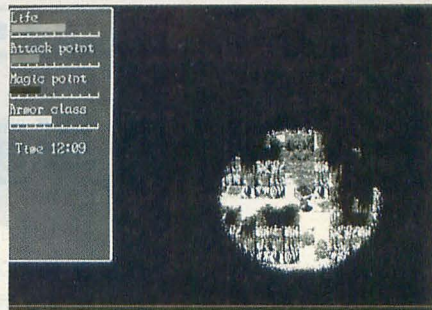
高橋「だって水の城に行ったら，そこの王様がドラゴンの角だか牙だかをほしいとか，ふざけたことぬかしやがるんだもん」

りゅう「そんな口きいてたら，牢屋に放り込まれるんじゃないの」

ここにドラゴンがいるらしいと聞きつけた我々はともかく地下の街への潜入に成功したのです。しかし，いかにも怪しげな街の倉庫には門番が立っていて，とても中には入れそうにありません。

高橋「うーん，困ったなあ」

りゅう「君は頭を使うのが役目なんだから，ちゃんと考えてよ」

まあ，中にはなんとか入れるでしょう。ただ，倉庫の奥にある洞窟に入るときはきちんとランプとオイルを買っておきましょう。ないと地獄を見ます。私は一度見ました。

まだまだ冒険は続くけど…◆◆◆◆◆◆

ドラゴンを倒したあともまだまだ冒険は続きますが，このへんで。

さて，気になる移植の出来のほうですがおおむね「良」といえそうです。X68000用にグラフィックは全部描き直してあるし（店の中などの人物の絵はちょっと浮いてるけど），デモのときに出てくる「フェアリーランド全景」(要するに広告の絵なのですが）は細部まできっちりと取り込み修正してあって思わず「おお！」という出来です。画面構成もウィンドウを使った新しいものになっていますしね。ゲーム中，キャラの動きがちょっとぎこちないのが残念ですけど。昔のゲームということで，そこここに「経験値稼ぎ」や「ちょっと不条理なんでないこの謎は？」が転がっていますが，それも「ああ，懐かしいな」という感じで，かえって好意的に受け止められます（少なくとも私はそうだった）。

「“ハイドライド”なんて知らないよ」という，いまどきのX68000ユーザーの方々はもちろん，すでにX1などでやった人も昔を懐かしんでもう一度プレイするのもオツなものだと思います。今回はTAKERUで販売ということで，比較的お手軽な価格ですしね。

音楽も忘れちゃいけないぞ

ハイドライド3は音楽も結構いけるのです。サンプリングなど派手なことはしてないし，ちょうどイースなどの陰に隠れてしまったこともあって，あまり取りざたされたことはありませんが，面白い曲が多いと思うのです。僕はこれのアレンジ版CDをいまでもよく聴いてますしね。お金に余裕がある方はこちらもどうぞ。

総合評価	0 ― 5 ― 10
アクション要素	★★★★★★★★
RPG要素	★★★★★★★
移植の出来	★★★★★★★★★
操作性	★★★★★★
8ビット時代の名残	★★★★★★★★

万人向けの,手軽で高品質なRPG

Urakawa Hiroyuki
浦川 博之

中世ヨーロッパを舞台にしたRPG。ショートシナリオタイプなので飽きずにプレイできる。この作品にはシナリオ5本が用意されているが,別にシナリオ集も発売されているので,終わった方はそちらもどうぞ。

よく考えれば,ひさびさのフィールド型のRPGの登場である。まだPC-8001が現役のころは,5×5マスしか表示されないマップの中を歩きまわって,ところかまわず戦って,いちばん強い奴を倒せばエンディングという“RPG”があっちこっちの雑誌に載っていた。いま,X68000からパソコンに入ってきた人には,この画面でアクション要素がないのが奇異に映るんだろうなぁ。時代も変わったもんだ。はあ。

おっと,思わず感傷に浸ってしまった。このブルトン・レイは,古き良き時代のRPGのスタイルを思い起こさせながら,実はそのシステムの品質は時代の最先端をいく,「自分が動かなければ敵も動かない」普通のRPGである。アクティブRPGではないぞ,念のため。

ゲームのシステムは◆◆◆◆◆◆◆◆◆

ゲームの進行に関しては従来のRPGとなんら変わるところはない。オープニングでストーリー説明があり,あとはそれをふまえて人の話を聞けば,やるべき事と会うべき人がわかるようになっている。武器と道具をそろえたら町の外へ旅立つという,お馴染みのパターンだ。

最近は,話の展開をスムーズにするためにマップを一本道にして次の目的地がすぐにわかるようにするのが流行だが,ブルトン・レイは自分で目的地を探し当てなければならない。マップのサイズそのものは,そう大きくないが,プレイしていると非常に広く感じる。

システムは比較的オーソドックスだが,そのなかで特徴を挙げるなら,徹底した自動化である。シナリオによっては2人か3人でパーティを組むことになるのだが,主人公以外のキャラクターの行動はコンピュータが担当し,プレイヤーはほとんど手が出せない。わずかに仲間が使う魔法の種類と使う頻度を設定できるだけである。メンバーに細かい指図をするわずらわしさを感じることなく,ひとりで冒険しているときも,パーティを組んでいるときも同じ操作感覚で遊べるようになっている。仲間のキャラクターは多少向こう見ずなところはあるが,しっかり自分が相手する敵を決めてかかっていく。バカな思考ルーチンにありがちな,ひとりで十分な敵に全員が飛びかかっていくといったことはない。

さらに自分の行動も,隣の画面への移動と,敵との戦闘についてはコンピュータに任せることもできるのだ。移動の際は複雑な地形でもきれいにまわりこんで歩いていくし,思いがけず敵に囲まれても応戦しながら退却する。自動戦闘モードも,その辺をうろついている怪物相手ならば十分使いものになる。

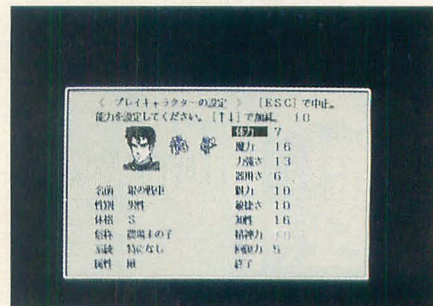
もうひとつ,プレイヤーの手間ヒマの浪費節減になってくれるのがオートマッピング機能。歩いたところをきちんと記録してくれる。PC-9801版ではシナリオ集の中で追加された機能だが,X68000では最初から装備している。これがあると,地図を完成させるつもりであちこち旅していれば,自然とゲームが進行してしまう。ほら穴のような見過ごしやすいものも,マップを見るとキチンと書かれているので,気づかなかったがためににっちもさっちもいかなくなるということもない。これがあるとないとではゲームに対する感想がだいぶ違う。オートマッピングが最初から使えるX68000ユーザーは素直に喜ぼう。

マルチシナリオだ◆◆◆◆◆◆◆◆◆◆

最近のRPG,特にアクティブRPGのシナリオはやたら大作指向だ。「世界の破滅を救うのは君だ!」「今解き明かされるナントカの謎!」,感動するまでは帰さんもんねという怒濤の張り手攻撃である。

ブルトン・レイのシナリオはシンプルで短い,童話や昔話程度の素朴なものだ。なんせブルトン・レイというタイトル自体,中世にイギリスからフランスブルターニュ地方にかけて流行った吟遊詩人の歌物語のことだそうである。まあそんなコンセプトだからして話の規模は小さいのも納得できる。その代わりに,5本のシナリオが用意されており,現在発売されているシナリオ集も合わせると30本以上のシナリオが遊べる。

また,ひとつのシナリオを終了したキャラクターを別のシナリオに転送することもできる。しかもそれはPC-9801用のユーザーディスクからであってもかまわない(ただし,キャラの顔が化けることがある)。さらにシナリオディスクまでPC-9801版と共用できるので,X68000用ブルトン・レイを手に入れさえすれば,シナリオ集は移植を待たなくても最新のもので遊ぶことができる。移植で待たされることの多いX68000のユーザーにとっては嬉しい話だ。その代わり,自分でシナリオをユーザーディスクに

X68000用 5"2HD版3枚組 8,800円(税別)
システムソフト ☎092(752)5278

キャラクター作成中。もっとタフにしたい

コピーするという，ちょっと面倒臭い手順を踏まなければならないが。

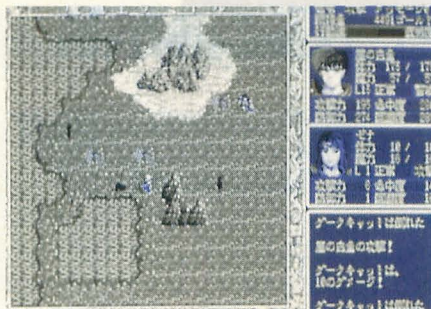
RPGとしてのルール◆◆◆◆◆◆◆◆

装備やステータスもなかなか充実している。むろんT&TやAD&DのようなヘビーデューティなRPGよりは簡素だが，シナリオのサイズには相応だ。

武器や防具にはS・M・L・LLの体格があって，体格の合わないものは使えない。またキャラクターの“力強さ”によって武器や防具をどのくらい持てるかが決まる。

戦闘能力は，攻撃力/命中度と防衛力/回避度の要素から成っていて，そのキャラクターの戦闘能力をちゃんと発揮させようと思ったら，キャラクターのレベルにあったものを用意してやらなければならない。たとえ宝箱から「炎龍の剣」なんて強そうな武器を手に入れたとしても，キャラクターに使いこなすだけの能力がなければ，かえってその重さのために防衛力/回避度が落ちてしまったりする。こういったキャラクターに対するまめなケアもRPGの楽しみのうち。武器屋に行くのが楽しくなろうというもんである。敵キャラクターにもこういった戦闘能力に基づいた個性があって，戦闘時にちゃんとお互いのステータスが生かされているのがわかる。

魔法は治癒に使うもの，相手を眠らせたり錯乱させるもの，相手の能力を下げるもの，攻撃を加えるものの4系統。それぞれ6種類ずつあって，合計24種類がある。もっとも，魔法がからきしダメなキャラクターでも，シナリオはクリアできるからご心配なく。

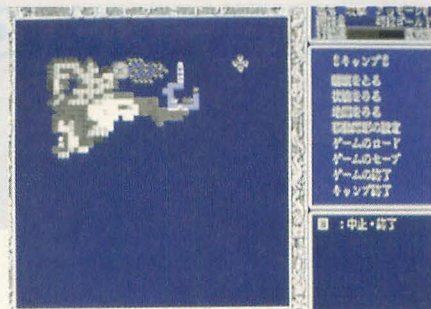
これらの特徴は，やはり複数シナリオ制のために生じたものが多いようだ。特定のシナリオに依存した演出やルールは作れないし，シナリオを作りやすくするために，ステータスその他のデータもわかりやすくまとめなければならない。斬新なところはないが，そのぶん自動化などのまっとうな部分を高品質に仕上げている。非常に洗練されてわかりやすいゲーム体系である。

メイン画面。地形も考えて戦おう

これが嬉しいオートマッピングだ

ゲームの評価は◆◆◆◆◆◆◆◆◆◆◆

X68000のRPGといえば海外の超大型RPGか，アクティブRPGしかなかったので，どちらにも属さないブルトン・レイは非常に新鮮に映った。シナリオは，短い代わりに適度なテンポで進んでいく。アクティブRPGの「壮大で一本道なシナリオ」とどっちがよいと聞かれたら，僕は迷わず前者を選ぶ。

べつにシナリオが短いだけで気に入っているわけではない。ブルトン・レイでは「西の湖に行ってみるとよい」といわれて，湖に向かって旅をしている間が楽しめるのである。自動移動モードを使いながら歩き回り，夜になったら寝る。敵が襲ってきたら戦う。この，当たり前の行動をしている時間のバランスがとれているせいだろう。無駄な手順を踏まされている気はしない。ストーリーでわざわざプレイヤーのやる気を誘わなくても，あちこち世界を見て回って，マップ作りを進めるだけでも十分楽しめる。だから話が単純でも構わないのだ。たとえあとで重要になる扉を先に発見してしまったとしても，ちょっとのぞいて帰ってくることが立派な冒険である。

よくRPGの弊害としていわれる経験値稼ぎもない。スタート時にゲームの難易度に合わせてキャラクターのレベルが設定されており，2・3レベルも上がると，その辺にいる怪物はゲームの障害ではなくなる。一定レベル以上でないと倒せないボスキャラもいない。経験値がゲーム展開の足かせになっていないのである。

反面ブルトン・レイの弱味を挙げるとすれば，プログラム的な質である。実はX68000版はPC-9801版の演出，画面をそのまま持ってきて極力同じにしてある。僕は高精細モードで画面を作るのには賛成だが，PC-9801で有効だった演出（たとえばラインスクロール）が，同じようにX68000ユーザーに通用するのかに関しては疑問が残る。確かに夜になったときの画面はPC-9801に比べてはるかに見やすいが，さらに目立つ部分でX68000ユーザーの優越感を刺激するような部分があってほしい。

また，PC-9801と比べてもパーティを組むようになると露骨に動きがカクカクしてくるし，マップを書き換えるときに時間を食ったりする。X68000ユーザーがこういうところに向ける目は，ものすごく厳しいのでちょっと心配だ。

しかし，それもこのゲームの魅力に比べれば微々たるものだ。遊べる時間やその質を考えればコストパフォーマンスはまちがいなく高い。コンピュータを始めたばかりで，RPGのイロハがよくわからない人，X68000用のゲームは過剰な演出が多くてという人には自信を持っておすすめできる。

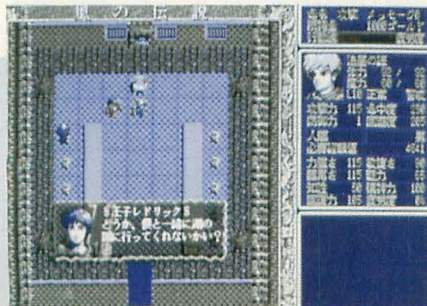
泣いて頼むんだったら行ってやってもいいぞ

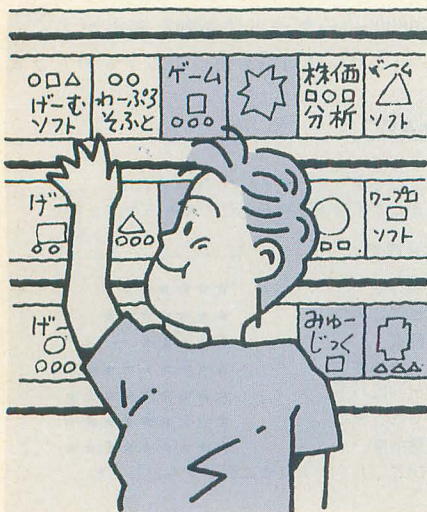
ぜーんぶまとめて

カクカクした動きを気にしなければ非常にいいゲーム。身構えて始めなければならない大作が多いなかにあって，「ちょっとやってみようかな」とふと手を伸ばす気になれるRPGは，ブルトン・レイくらいのものだ。マップをもうちょっと完成させてみようかなと歩き回っていれば，自然にキャラクターも育つしストーリーも進む。いつの間にかすっかりハマってしまっているというわけだ。ゲームはバランスだなあと思わせる一作。

シナリオ集には，ややレベルの高いものをおさめたVol.1と，ユーザーが作ったひとクセもふたクセもありそうなVol.2が用意されていて，これまたどんなドラマを見せてくれるか楽しみだ。

総合評価	0　5　10
操作性	★★★★★★
ビジュアル	★★★★★★★
音楽	★★★★★
シナリオ	★★★★★★★★
ゲームバランス	★★★★★★★★★
マニュアル	★★★★★★★★★
熱中度	★★★★★★★★★

＊ひとこと：シナリオエディタも出してね

AFTER REVIEW

諸々の事情でしばらくお休みしていた「AFTER REVIEW」ですが，今回から復活です。復活第1弾記念ということで(?)，今回はシムシティーだけをドンと取り上げます。発売と同時に飛ぶように売れたそうで，すぐには買えなかった人もいるかもね。

シムシティー

▶シムシティーは楽しんでやっているんだけれど，人口50万人など，とてもできない。39万人で止まっている。できるだけ陸が多いものを選んだつもりなんだけど……。

岡田　政弘(19)東京都

▶シムシティーの面白いところは，メガロポリスにすることよりは1歩手前で災害の連発をして，住人の逃げるところを見ることです。やはり，みんなも同じ考えでしょう。　土屋　義孝(20)神奈川県

▶私の街は，いま27万人ほどである。23万人を超えるあたりで，このままでは人口が増えないと悟り，大改造（？）をしているうちに，街がきらいになってやっていない。交通渋滞の解消のために，道路を公園にしてしまう自分がいやになった。

竹田　文義(25)静岡県

▶シムシティーにスイッチをつけて起動すると楽しい。コマンドモードで，

A>SIM/H	スゲー細かいモード
/HL	(24KHz)
/A	いきなり新しい町作り
/LJ	なぜかバスエラー
/LE	〃

最後のは日本語と英語の切り替えだと思うが……。　渡辺　篤志(17)滋賀県

▶シムシティー楽しいですね。でも，不満もあります。斜めの道が作れない！　地下鉄が作れない！　遊園地は？　ディズニーランドは？　都市高速道路がほしい。最後に「市役所はどこ？」でした。

魚住　雄一(21)愛知県

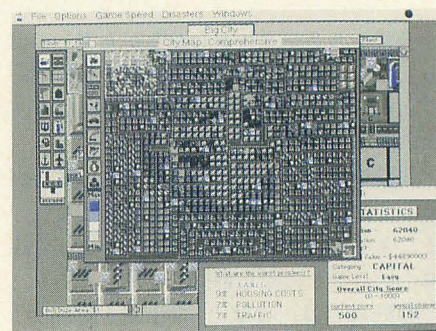
▶ポピュラスはプレイしている自分が多少醜くなる。シムシティーはプレイしている自分が知的に思える。

寺門　修司(19)兵庫県

▶怪獣はなかなか，かわいらしくていい。でも，にくい。　田中　哲(17)滋賀県

▶完璧に区画整理するよりも，住民の要求のままに拡張したほうが人口が増えるなど，妙に現実っぽいところがいい。

小川　英範(21)新潟県

▶街づくりが楽しい。しかし，曲がった道を見ると直したくなってしまう。石井　久(18)千葉県

“Big City”，メルトダウンしているぞ

▶下手な社会科の授業を聞くより，よっぽどためになるし面白い。思わず自民党に味方したくなってしまう。

手賀　寛(16)福井県

▶自分で作ってみたいと夢に見ていた街が作れるから。　岩崎　正道(16)熊本県

▶ぼーっと見ているだけで進んでいくのがいい。ついでに街を災害で壊すのも好き。アブナイ？　山本　昭治(22)神奈川県

▶基本的に勝敗がないので腹も立たない。しかし，この街の市長は不老不死か？　私なんぞ，かれこれ950年も生きているぞ。

岡田　伸一(22)京都府

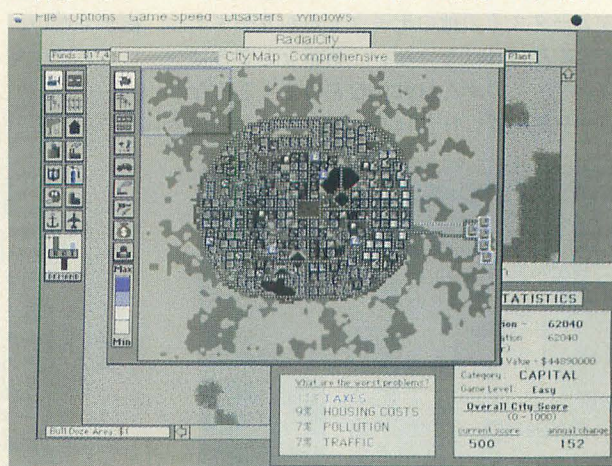
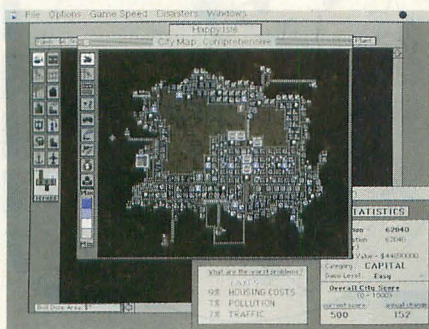
▶交通機関は鉄道だけに命をかけて，発電所はすべて原子力発電所。なんかあぶないですね。　水口　仁郎(21)岡山県

▶自分の作っている街がしだいに大きくなって，車なんかがたくさん走っていくようになっているのを見るとなんとなくいい。

紺谷　健(20)石川県

▶土木学科で都市計画をやっているので。

和田　安生(22)福岡県

▶人間の社会生活が営まれるうえでの集団心理の重大さと，人間の環境への依存性が

“Radial City”，計画的に，放射状に

"Happy Isle", 本当に幸せな島かな？

"Linear City", 結構ありがちかな

よくわかるから。　原田　真志(20)静岡県

▶アイデアとゲームにする際のモデル化がすばらしい。　橋本　忍(19)埼玉県

▶試験が近かったのに，誘惑に負けてシムシティーを買ってしまいました。家に帰ってマニュアルとにらめっこしながら，おそるおそるやっていたところ，突然「メルトダウン」しました。初めてパッケージを開けてから20分後のことでした。誰かこの記録に勝てる人はいませんか。

村上　淳一(19)福岡県

▶神様はちと荷が重すぎるので。

志賀　宗一(17)愛知県

▶日本風にアレンジしたバージョンがほしい。山とか田んぼとかあって，埋め立てができて，地上げ屋がいて，峠に道を作るとなぜか車が集まるとか（笑）。

鹿又　健(21)栃木県

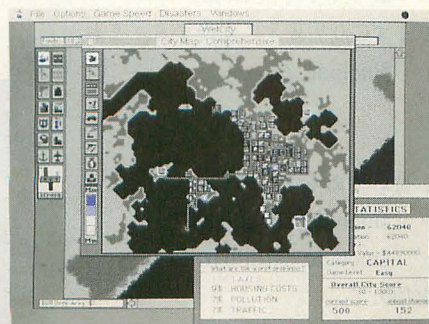
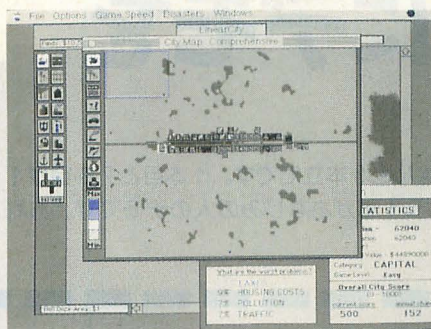
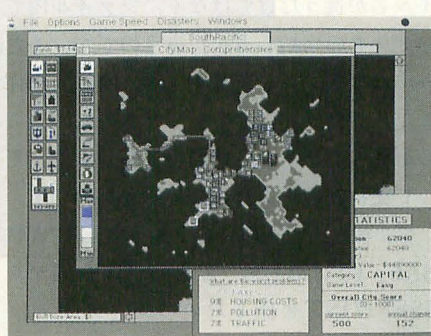
▶シムシティーの対戦版ってないかな……。ひとりは普通にやって，もうひとりは怪獣を出したり，飛行機を落としたり……。バンゲリングベイみたい。

伊藤　一範(18)三重県

▶ゲームというメディアに対する，一種の思想が感じられるから。

角倉　裕之(21)鹿児島県

▶とても面白い。エンドレスなところがいい。　黒須　政広(22)愛知県

▶市長気分を味わえるから（本物よりかなり大変そう）。　前田　秀樹(16)京都府

"Wet City", 人口増加に不利な地形

"SouthPacific", その名にふさわしい

▶シムシティーじゃプレイヤーは市長ってことになっている。でも，市長というよりはなんちゅーか都市計画屋さんの気分だな。緑を増やして，治安をよくして，ゆとりある交通をと。「うーん，こりゃいい街並だ。増税も通しだな」なんて悦に入っているのに，ちょっと税率を上げると，もう住民のうるさいこと。税金が高いだの，地価が高いだの，まったくこいつらときたらものの価値を解さないんだから。

ま，一人よがりは禁物ってこと。100年以上もの時間をかけて都市とともに生きるのだ。その間，都市が成長する過程で住民の価値感は変わっていく。そう，工場が町の希望だった時代から，エコロジストがはびこる時代へと。これがシムシティーのすごいところだな。時代ごとにセーブしてあとで歴史を振り返ってみるのも一興かも。

志村　一(27)東京都

シムシティーに対するハガキは山のように多く，さすがに名作という感じですね。また，ポピュラスと比べる人もかなりいて，やはり永遠のライバルというところでしょうか。ちなみに回りの写真はイマジニアからいただいた，作者のウィル・ライトさんが作った街です（Mac版）。

X68000用　5″2HD版　9,800円(税別)
イマジニア　☎03(343)8911

発売中のソフト

★レインフォーサー　ザインソフト
X68000用　5″2HD版4枚組　8,800円(税別)
★ダイナマイト・デューク　ヘルツ
X68000用　5″2HD版3枚組　8,800円(税別)
★ザ・スーパーラスベガス　日本デクスタ
X68000用　5″2HD版2枚組　12,800円(税別)
★ブルトン・レイ　システムソフト
X68000用　5″2HD版3枚組　8,800円(税別)
★ソル・フィース　ウルフ・チーム
X68000用　5″2HD版3枚組　8,800円(税別)
★銀河英雄伝説II　ボーステック
X68000用　5″2HD版4枚組　9,800円(税別)
★栄冠は君に　アートディンク
X68000用　5″2HD版3枚組　9,500円(税別)
★イメージファイト　アイレム
X68000用　5″2HD版2枚組　9,700円(税別)
★続ダンジョン・マスター　カオスの逆襲
ビクター音楽産業
X68000用　5″2HD版2枚組　8,800円(税別)
★リングマスターII　ホビージャパン
X68000用　5″2HD版　8,800円(税別)
★スペースローグ　ウェーブトレイン
X68000用　5″2HD版　9,800円(税別)
★D～欧州蜃気楼～　ウルフ・チーム
X68000用　5″2HD版3枚組　12,800円(税別)
★シュヴァルツシルト　工画堂スタジオ
X68000用　5″2HD版2枚組　12,800円(税別)
★エメラルドドラゴン　グローディア
X68000用　5″2HD版6枚組　9,800円(税別)
★ぴくせる君　ブラザー工業
X68000用　5″2HD版　4,800円(税別)

新作情報

★DRAKKHEN　EPIC/SONY RECORDS
X68000用　5″2HD版　価格未定
★ワールドスタジアム　SPS
X68000用　5″2HD版2枚組　8,800円(税別)
★マーブルマッドネス　ホームデータ
X68000用　5″2HD版　価格未定
★ファンタジーIV　スタークラフト
X68000用　5″2HD版　9,800円(税別)
★中華大仙　シャープ
X68000用　5″2HD版　価格未定
★エイリアンシンドローム　電波新聞社
X68000用　5″2HD版　価格未定
★アトミック・ロボキッド　システムサコム
X68000用　5″2HD版2枚組　8,800円(税別)
★プリンス・オブ・ペルシャ　ブロダーバンドジャパン
X68000用　5″2HD版　価格未定
★パロディウスだ！　コナミ
X68000用　5″2HD版　価格未定
★生中継68　コナミ
X68000用　5″2HD版　価格未定
★遥かなるオーガスタ　ティーアンドイーソフト
X68000用　5″2HD版2枚組　12,800円(税別)
★ラプラスの魔　エム・エー・シー　ハミングバード
X68000用　5″2HD版3枚組　価格未定

REVIEW原稿を送っていただいた志村さんにはOh!X特製記念品をお送りします。

★(で)のショートプロぱーてぃ その16

クリスマスにデモを

Komura Satoshi 古村 聡

クリスマス近し，ということで今月はなんと3本立てです。1本目と2本目が1組でX68000用外字ユーティリティ，もう1本がおなじみの人のデモです。あれれ，実質的にはふたつということかな。

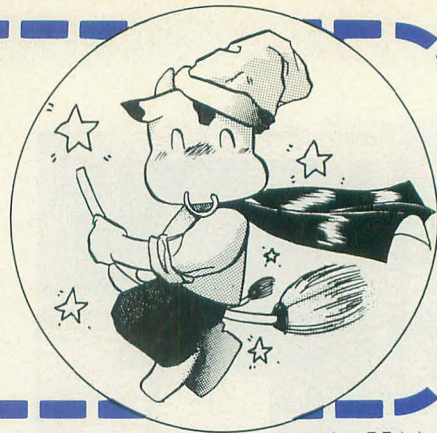



illustration : T.Takahashi

どもどもども。でへへ。私が恐怖の宴会男（で）であります。

いやあ，飲んだ飲んだ。実はライター同士の飲み会があって，イラストでおなじみの高橋君やあの“CARD.FNC”の毛内さんたちと飲んできたんですよ。ああ，気持ちいいな。どっからでもかかってきなさい。私は誰のイッキでも受ける！

突然，話は変わりますが（こればっかし），せっかく足を洗いかけていたのに，また少女まんが買っちゃったんです。わかつきめぐみさんの“So What?”というやつ。本屋で1冊ぱらぱらっとめくってみたら面白そうで，気がついたら某新宿の某に行って（実は初めて）残りの2巻から6巻まで買ってしまった。せっかく買わないようにしていたのに……。ということで，なんかほかに面白い少女まんがないですか？ あ，ついでにわかつきめぐみさんがほかにどんなのを描いてるか教えてくれるのもうれしいです。

外字の使い回し

では，たまには真面目にお仕事でもしましょうかね。今月の1本目と2本目は外字ユーティリティ「USK2WP.BAS」，「WP2USK.BAS」です。

USK2WP.BAS & WP2USK.BAS for X68000 (X-BASIC)
東京都 木村哲也

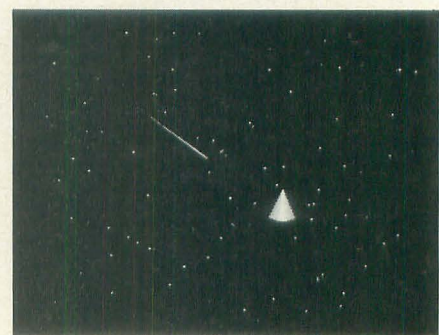
この2本のプログラムは「USKCG.SYSとワープロのユーザーフォントとの外字データを変換する」ためのユーティリティです。8月号でもあったようにUSKCG.SYSには割といろいろな外字がありますが，これらはワープロでは使えません。単にファイル名が違うだけではなく，ファイル内でのデータの格納の方法が違うのです。また，外字はワープロで定義すると，24×24を作るだけで16×16のものもできるので，こちらを使ったほうがいい場合もあります。そこで便利なのがこれらのツールなのです。

で，使い方。その名のとおり「USK2WP.BAS」（2はtoと思ってね）がUSKCGからUSERFONTを作るためのプログラム，「WP2USK.BAS」がUSEREFONTからUSKCGを作るためのプログラムです。

まず，カレントディレクトリにUSK2WPのときはUSKCG.SYSを，WP2USKのときはUSERFONT.DATを置いてRUNしてください（ファイル名が自動的に指定されますのでカレントにUSKCGあるいはワープロのUSERFONTがないときは，“！ドライブ（ディレクトリ）名”でカレントを移動するようにしてください）。無事に，「終了しました」との表示が出たらデータが変換されています。

おっと，注意。このプログラムはバックアップファイルを作成するといったことはしていないので取り扱いには十分注意してください。

出たな，BASICを使った実用ユーティリティ。私はうれしいぞ。そう，「BASICという言語はスピードさえ気にしなければ，結構いろいろと作れるものなのであります」と思っている私としては非常にうれしいことであります。コンパイルしちゃえばスピードも気にしなくっていいということでX-BASICってのはおいしいですよね，本当に。

ただ，私はWP.Xは使えない人なので（普段からASKにムチャクチャなキー割り付けをしているのでWP.Xは手が受けつけてくれないのだ）ちょっと残念だったりもするのだけど，WP.Xを使っている人にはきっと便利なプログラムに違いあるまい。うむうむ。

COMET

そうそう，Cコンパイラを持っている人はHuman68kの外部コマンドにしてみるときっと便利でしょうね。そうすれば，BASICを立ち上げなくても使えますし，FORMATやDIRなんかと同じ感覚で使えますからね。え？ BASICで書いたものをコンパイルすると，画面モードが512×512になるし，ファンクションキーも変わるから，BASICで書いたっていうのがモロにばれちゃってほかの外部コマンドと同じように使えない？

それにはですね，

```
bc　ファイル名.bas
```

でまずCのリストに変換する。それから，

```
#include <……>
    ⋮
    ⋮
```

と書いてあるところに，

```
#include <doslib.h>
#include <stdlib.h>
```

を加えて，

```
b_init()；
```

を削る。そして，

b_exit(0)；

を，

C_CURON()；

exit(EXIT_SUCCESS)；

に書き換える。早い話，512×512ドットモードの画面にする部分と，そこから戻る部分を取っただけなんだけど，これで一応，「Cで書いたんだもんね，どうだすごいだろう」という顔ができます（おいおい）。ちょうどいいサンプルもあるし，XCを買った方はぜひとも試していただきたいな，と思う（で）なのでありました。じゃんじゃん。

あ，そうだ。XCver.2.0を使ってコンパイルした場合はwarningが大量に出ますがこれは気にしないでそのまま使ってOKです。

さあ，この手でBASICでガシガシと外部コマンドを作ってしまおう。そしてショートプロに送るのだ。

もうすぐクリスマスなのね

さてさて，では今月ラストの3本目。もう常連になってしまった太田さんの作品で，X-BASIC用デモプログラム「COMET.BAS」です。

COMET.BAS for X68000

(X-BASIC)

東京都　太田敬三

太田さんの投稿原稿曰く，「これは自信ないなあ。でも友達が面白いと言ってくれたので送ろう。フロッピーディスクは1Mバイトもあることだし。ぽん」。

簡潔なご説明ありがとうございました。実はこれ，一緒に送られてきたほかのプログラムが投稿用のプログラムでこのプログラムはおまけだったのですね。まあ，世の中とは得てしてこんなものなのでありました。うむうむ（選んでる私の性格が悪いだけという話も……）。

おっとプログラムの話ね。

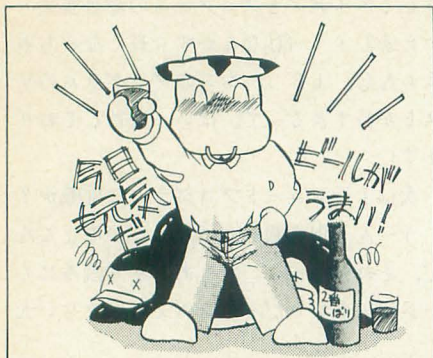
画面が星空になり三角錐が現れ，そして彗星がやってきます。皆さんにはぜひ部屋の電気を消して見ていただきたい。美しい……。ううっ。デモプログラムってどうしても説明が短くなっちゃうんだよね。あんまり説明しちゃうと打ち込んだときの感動が半減しちゃうし……。これじゃ，おまんまの食いあげだよ。うるうる。

いやそれにしても，なんか前々作の「夜見てはいけないデモ」のときも前作の「かべくずし」のときも言ったような気もするのですけど，これだけ短いプログラムでこんなものを作ってしまう太田さん，さすがですね。しかも，今回にいたってはBASICの1画面プログラムですからね（WIDTH96だとぴったり1画面におさまる）。たいしたもんです。読みやすいプログラムのまま1画面でおさめてしまうのはまさしくアイデアとプログラムの総合勝負といえましょう（私なんかすぐ長くなっちゃうもんな。え？　ハンズのプログラムのリストが長すぎるって。はい，反省しております）。

次は1行ショートプログラムに挑戦か？

うーん，何本載せればページが埋まるんだ，それって……。でもあんまり読みにくいショートプロだったらちょっとぐらい大

リスト1　USK2WP.BAS

```
 100 /* FROM USKCG TO USERFONT(WP)
 110 /*                    TRNS UTILITY
 120 int ch,fp1,fp2
 130 dim char lim(2),lin(74),nl(74)
 140 str fname1,fname2,strbuf
 150 fname1="USKCG.SYS"
 160 fname2="USERFONT.DAT"
 170 fp1=fopen(fname1,"r")
 180 fp2=fopen(fname2,"c")
 190 if fp1<0 or fp2<0 then print"エラーが発生しました"
 200 while(fp1>=0 and fp2>=0)
 210 print"USKCG.SYSをWP.Xの外字データに変換します"
 220 print"何かキーを押してください"
 230 strbuf=inkey$
 240 print"作業中です"
 250 /*      最初 34バイト 書き込む
 260 fread(lin,34,fp1)
 270 print".";
 280 /*      FF FF がでるまで16×16を読み書き
 290 while not feof(fp1)
 300  fread(lim,2,fp1)
 310  ch=ch+1
 320  if lim(0)<>255 and lim(1)<>255 then fread(lin,32,fp1) els
e break
 330  fwrite(lim,2 ,fp2)
 340  fwrite(lin,32,fp2)
 350 print".";
 360 endwhile
 370 /*      余った部分に 00 を書き込む
 380 disk(&H76,ch,32)
 390 disk(&H77,1,32)
 400 print".";
 410 /*      FF FF を書く
 420 fwrite(lim,2,fp2)
 430 /*24×24(74ﾊﾞｲﾄ)を16×16と同じ数だけ読み書き
 440 for i=1 to ch-1
 450  fread(lin,74,fp1)
 460  fwrite(lin,74,fp2)
 470 print".";
 480 next
 490 /*      余った部分に 00 を書き込む
 500 disk(&H76,ch,72)
 510 disk(&H77,1,72)
 520 /*
 530 fclose(fp1)
 540 fclose(fp2)
 550 fp1=-1
 560 print
 570 print"終了しました"
 580 endwhile
 590 end
 600 /*
 610 /*
 620 func disk(x1,x2,x3)
 630  lin(1)=x1
 640     for i=x2 to 94
 650         lin(0)=&H21+i-1
 660         fwrite(lin,2,fp2)
 670         fwrite(nl,x3,fp2)
 680     next
 690 endfunc
```

きくても読みやすいプログラムのほうを掲載することが多いから気をつけてね。

三角錐に向かって飛んでくる彗星の軌道がキュベレーのファンネルみたい。行け，当たれ，ビュンなんてね。それにしてもきれいですね。星空に彗星なんて，クリスマスが近づくいまの季節にぴったりじゃありませんか，ロマンチックで。

ああ，もうすぐクリスマスか。クリスマスといえば……，飲み会！　そのあとは忘年会，その次が新年会じゃないか。飲んでばっかりでうれしいなっと。あ，なんか頭痛くなってきた。二日酔いかな。

じゃ，そういうことでまた来月。

★動かないよの部屋（突然質問箱）

Q. 9月号のショートプロぱーてぃに載っていた「なさけない星」をGCCでコンパイルしたところ，線が1本しか出なんです。何度も打ち間違いがないか確かめました。そして，よくわからんとあきらめたのです。2，3日して，突然，XCでコンパイルしてみたらいいのでは，と思って早速試してみたらうまくいきました。結局GCCだと速すぎたということなんでしょうか。

A. 実を言うと私もどう説明していいのかわからないんですが，とりあえず説明してみます。ラインは本当に1本しか描いていません。というのも，このプログラムではrnd( )関数を使って円を描くように少しずつ0度から360度まで角度を変えて線を引いているんですが，そのリスト中の，

rnd ( ) ＊10

という部分で，0から10未満の値が返ってくるはずなのに，GCCでは，

21299990528

という，よくわからないばかでっかい数値が返ってきてしまうのです。(XCのライブラリマニュアルによればrnd関数は0から1までの範囲で乱数を返す関数です)。で，1本線を引くと次が360度を越えてしまうのでもう線が引かれない，……つまり線が1本しか引かれないのです。

また，XCでもver.2.0になるとGCCと同様になり，線は1本しか引かれません。これはどうも変数の型の違いによるもののようです。GCC，あるいはXCのver.2.0をお使いの方は，インクルードするもののなかに，rnd( )が定義してある，

#include <basic. h>

を付け加えるとうまくいくようです。

リスト2　WP2USK.BAS

```
100 /* FROM USERFONT(WP) TO USKCG
110 /*             TRANS UTILITY
120 int fp1,fp2
130 dim char lin(74),nl(74)
140 str fname1,fname2,strbuf
150 fname1="USERFONT.DAT"
160 fname2="USKCG.SYS"
170 print"WP.Xの外字ファイルをUSKCG.SYSに変換します"
180 print"何かキーを押してください"
190 strbuf=inkey$
200 print"作業中です"
210 fp2=fopen(fname2,"c")
220 fp1=fopen(fname1,"r")
230 if (fp1<0 or fp2<0) then print"エラーが発生しました"
240 while(fp1>=0 and fp2>=0)
250 /*   最初の34バイト読みとばす
260 fwrite(nl,34,fp2)
270 /*   16×16（34バイト）94回読み書き
280 for i=1 to 94
290  fread(lin,34,fp1)
300 print".";
310  fwrite(lin,34,fp2)
320 next
330 /*   FF FF がでるまで読みとばす
340 while lin(0)<>255 and lin(1)<>255
350  fread(lin,2,fp1)
360 endwhile
370 /*   FF FF を書き込む
380 fwrite(lin,2,fp2)
390 /*   24×24（74バイト）94回読み書き
400 for i=1 to 94
410  fread(lin,74,fp1)
420  fwrite(lin,74,fp2)
430 print".";
440 next
450 /*
460 fclose(fp1)
470 fclose(fp2)
480 print
490 print"終了しました"
500 endwhile
510 end
```

リスト3　COMET.BAS

```
   10 screen 0,2,1,1:randomize(val(right$(time$,2))*123)
   20 int nx(6),ny(6),cd(5),c=1,no,dx,dy,vx,vy,mx,my,zx,zy
   30 apage(1):for i=0 to 99:pset(rnd()*256,rnd()*256,rnd()*256)
:next:apage(0)
   40 while 1
   50   zx=abs(mx-dx):if zx>8 then zx=8
   60   zy=abs(my-dy):if zy>8 then zy=8
   70   vx=vx-(vx<-zx)+(vx>zx)-(dx>mx)+(dx<mx)
   80   vy=vy-(vy<-zy)+(vy>zy)-(dy>my)+(dy<my)
   90   mx=mx+vx:my=my+vy
  100   line(nx(no),ny(no),mx,my,c)
  110   no=no+1:if no=7 then no=0
  120   i=no+1:if i=7 then i=0
  130   line(nx(no),ny(no),nx(i),ny(i),0)
  140   nx(no)=mx:ny(no)=my
  150   for i=0 to 5
  160     palet(c,cd(i))
  170     c=c-1:if c=0 then c=6
  180   next
  190   c=c+1:if c=7 then c=1
  200   if mx-dx or my-dy then continue
  210   apage(1)
  220   a=rnd()*192:for i=0 to 5:cd(i)=hsv(a,31,31-i*5):next
  230   for i=0 to 30:circle(dx,dy,27,0,-240-i,-300+i,400):next
  240   dx=rnd()*216+20:dy=rnd()*226:a=8*int(rnd()*31)+8
  250   for i=0 to 30:circle(dx,dy,27,a+i/4,-240-i,-300+i,400):n
ext
  260   apage(0)
  270 endwhile
```

(で)のぱーてぃハンズ第2部――(その2)

どもども，お待たせせいたしました。こちら，謎の大帝国ぱーてぃハンズでございます。

ほい。先月号でダンジョンの壁のパターンが描けたんで今度はこいつを使って少しずつ迷路プログラムっぽくしていくんですね。先月号でいうと，

“2) 自分から見える壁を選んで1）に壁を描かせるルーチン”

の部分を作っていくわけです。うーん，ボトムアップ（といいつつリストはトップダウンくさかったりするけど，気にしないでね)。

さてさて，それでは自分から見えるっていうのはいったいどういうことなんでしょう。さて，いま何が見えてますか？　私は……（あ，編集さんの怖い顔）目をそらしてと。ついでにバリケードを構築して……。はいはいっ，いますぐ原稿あげますからっ，バリケードを越えてこないで！　あははは。つまり，ものが見えるには，

「自分の視界の範囲にある」

「じゃまものに遮られていない」

というふたつの条件が必要になってくるわけですね(まあ，強引)。まず最初に，“自分の視界”なるものを作っていきます。はい。

目の前のものは?

さてさてさて。無限の彼方まで360度展開で見えるととってても面白いんだけど，世の中そうはいかない。ましてや今回のプログラムはダンジョンのパターンをすべて手で描き出してしまったのでそんなことはのぞむべくもないというわけです。

そんでですね。先月の図2を引っ張り出してきて……。自分からどの範囲のものが見えるんでしたっけ？　まず，自分の前と左右は見える。で，ひとつ，ふたつ……先は4つまで見えるんですよね（サイドの壁は3つだけどこちらを向いている壁が4つ先まであるでしょ)。ということは，4×3の大きさのマップを作ってやればいいわけですね。つまり，これが自分の視界ということになるわけです。

配列なのである

さて，と。今度はマップを作るわけですね。でも，これは方眼紙ではなくてなんとかしてプログラム上にマップを表現しなくちゃいけない。マップを表現するのに都合のいいもの。変数があって，それがたくさんつながっていて……。そう，配列なんです。整数の配列を作ってその中に1とか0とか書いて，そこが空いているとか壁になっているとかを表すようにすればいいのです。

ここでは4×3のマップなので，

```
int ary(11) = {
    ……
}
```

として，整数型の変数を12個くっつけた1次元配列を作ります。4×3なんだから2次元配列で（4，3）にすればいいじゃないかっていう声も聞こえてきそうですがそこは私の趣味です。気にしないように。

んで，配列の添字，つまり，ary(1)とかの1の部分とマップ上での位置との対応なんですが，

```
0  1  2  3
4  5  6  7
8  9  10 11
```

(数字が配列の添字。位置がマップ上に対応)というふうにします。つまり自分はary(4)にいて自分のすぐ右がary(8)，左はary(0)で目の前がary(5)，その先がary(6)，ary(7)ということになるんです。

さて，ガンガン進めていきましょう。入れものが決まったら，次は中身。

配列が決まったわけですけど，整数の配列ですから32ビットの範囲内ということで268435455(日本全国民に1人ひとり番号をふっても，まだ半分……）種類のものが入れられますね。いろいろなイベントをここで定義できるわけです。たとえば上に通じる階段とか，魔物の集会場とか。

ここではとりあえず，

0……何もない

1……壁

2……赤い壁（通り抜けられる）

の3つを決めておきます。でね，実際に絵を描くときには，

0……何も描かない

1……白で壁を描く

2……赤で壁を描く

と絵を描くときに色分けするようにしてあげてるんです。

つまり，“そこにあるものの種類から壁の色を返す”関数を作る必要があるわけですね。

よし，マップに関しては決まったと。あとは描いていくアルゴリズムだな。

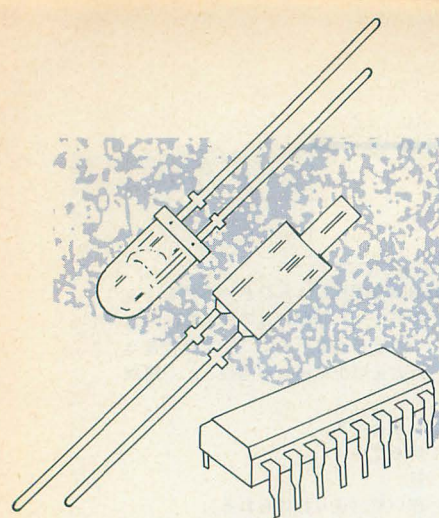
来月につづく

……と思ったら，もうページがないじゃないか！　今月もリストが長いしなあ。1ページを越えちゃうとなんとなくまとまりがないし……，うるうる。壁を描くアルゴリズムの解説，そしてプログラムを作るところまでいこうと思ってたのに，なんてこったい。

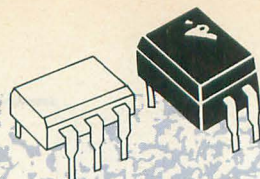
ということで，掲載するプログラムのほうは来月分のマップ（配列）上のものを見て壁を描くところまで載せておきます（だってもったいないんだもん)。で，来月はこの壁を描くアルゴリズムとプログラムの解説をしていきます。それまで配列にいろいろ入れて遊んでみてくださいな。

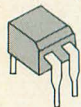
では，来月またこのOh！Xで。つづくっ！(と看板を立てて去る)

リスト

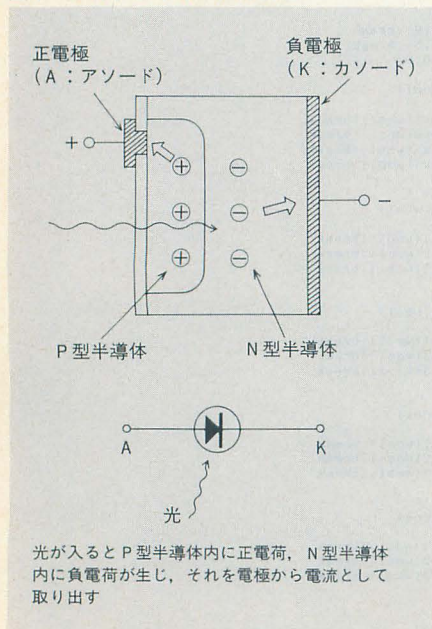
```
  10 /*変数のイニシャライズ*/
  20 int i,c=3,myxy=22,dir=2,mflg
  30 str a
  40 /* x=          0 1 2 3
  50 int ary(12 )={1,0,0,1,
  60               0,0,0,1,
  70               1,0,0,1 }
 305 drawmaze()
 380 end
1880 func drawmaze()
1890 /*迷路を描く(自分のX, Y、X, Yの増分値)/
1900 int i,j,k
1910 fill(0,0,383,383,0)
1920 i=0:j=0:wkin=0
1930 for k=4 to 7      /*まず中央についてみる*/
1940      if j=0 then i=i+1
1950      if ary(k)<>0 then j=i:if wkin=0 then wkin=ary(k)
1960 next
1970 if j=0 then i=5
1980 wcol=wkind2wcol(wkin)
1990 if i<5 then drawbox(i,wcol)
2000 /*次に左*/
2010 i=i-1:j=0
2020 while(j<i)
2030 /*print j                           */
2040 if ary(j)<>0 then drawsidel(j+1,wkind2wcol(ary(j))) else i
f ary(j)=0 and ary(j+1)<>0 then drawnol(j+2,wkind2wcol(ary(j+1))
)
2050 j=j+1
2060 endwhile
2070 j=0
2080 while(j<i)      /*そして右*/
2090 if ary(j+8)<>0 then drawsider(j+1,wkind2wcol(ary(j+8))) el
se if ary(j+8)=0 and ary(j+9)<>0 then drawnor(j+2,wkind2wcol(ary
(j+9)))
2100 j=j+1
2110 endwhile
2120 endfunc
2130 func wkind2wcol(kind) /*マップ上の記号と壁の色の対応*/
2140 switch kind
2150      case 1:return(9):break
2160      case 2:return(5):break
2170      default:return(0)
2180 endswitch
2190 func drawbox(depth,wcol)
2200 switch depth
2210      case 1:drawbox1(wcol):break
2220      case 2:drawbox2(wcol):break
2230      case 3:drawbox3(wcol):break
2240      case 4:drawbox4(wcol):break
2250 endswitch
2260 endfunc
2270 func drawsidel(depth,wcol)
2280 switch depth
2290      case 1:drawsl1(wcol):break
2300      case 2:drawsl2(wcol):break
2310      case 3:drawsl3(wcol):break
2320 endswitch
2330 endfunc
2340 func drawsider(depth,wcol)
2350 switch depth
2360      case 1:drawsr1(wcol):break
2370      case 2:drawsr2(wcol):break
2380      case 3:drawsr3(wcol):break
2390 endswitch
2400 endfunc
2410 func drawnol(depth,wcol)
2420 switch depth
2430      case 2:drawnl1(wcol):break
2440      case 3:drawnl2(wcol):break
2450      case 4:drawnl3(wcol):break
2460 endswitch
2470 endfunc
2480 func drawnor(depth,wcol)
2490 switch depth
2500      case 2:drawnr1(wcol):break
2510      case 3:drawnr2(wcol):break
2520      case 4:drawnr3(wcol):break
2530 endswitch
2540 endfunc
```

ハードウェア工作入門《7》

センサー回路その1

Misawa Kazuhiko
三沢 和彦

今回からA/Dコンバータを利用した身近でより実用的な応用例に挑戦することにします。取り上げるテーマはセンサー回路です。まず1回目は理論編ということでセンサーの種類や具体的な働きを整理することから始めてみましょう。

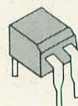
前回までにA/Dコンバータボードを完成させ，ごく簡単な応用としてアナログジョイスティックを試してみました。しかし，それだけでは，A/Dコンバータの本領を発揮させていません。今月からは，A/Dコンバータの本格的な応用として，いろいろな種類のセンサーをこのA/Dコンバータボードに接続してみることにします。

センサーとは何か

10月号のA/Dコンバータの解説の中で，機械が人間の感覚の代わりに自然界の量を計測するシステムについて述べました。そして，このようなシステムにおいては，光の明暗，音の強弱，温度の高低などのアナログ量は，すべて電圧に変換されたのちにA/Dコンバータに入力されます。この，自然界のアナログ量を電圧に変換する装置をセンサーといいます。

センサーはもともと人間（や他の生物）の感覚器官を代替し，その機能を拡張したものを意図して作られましたから，基本的なセンサーとして，人間の五感（視覚，聴覚，触覚，味覚，嗅覚）に対応したものがあります。しかし，人間の感覚には，自然界の物理量を「感知する」ことに加え，感知した信号から外界の対象についての情報を「認識する」機能があります。

たとえば，視覚に関していえば，目に入ってきた光が網膜上に焦点を結ぶとき，網膜上の各点における細胞が光の色と強度とを検出する段階が「感知」にあたります。その次の段階として，網膜上の各点で感知した信号を組み合わせ，それを像として捉えることによって，対象の姿や形，または位置関係を把握する段階が「認識」ということになります。

人間の感覚器官を機械的または電子回路的にシミュレートできるのはせいぜい「感知」の段階までで，「認識」の段階まで行うのには限界があります。現在「感知」の段階であれば，実用上で人間より優れた性能を示すセンサーは多くあります。視覚では光センサー，聴覚では音波センサー，触覚では圧力センサー，味覚と嗅覚ではガスセンサーなどが対応します。要するに，人間の持つ感覚機能のなかで，センサーは特定の物理量だけについて精密に計測する機能にのみ対応しているというわけです。ですから，実際の感覚機能をシミュレートするには，複数のセンサーによってより多くの物理量を計測し，それらの情報をコンピュータで処理しなければなりません。このとき，コンピュータは人工知能のシミュレートをしていることになります。

現在のセンサー技術においては，特定の物理量を精密に計測する性能は特に優れています。人間の感知できる光や音波であっても，感知可能な領域は限られていますが，センサーであれば，人間の感知できない紫外・赤外光や超音波でも高感度に検出できます。また，人間がまったく感知できない磁気や放射線などのセンサーもあります。

そこで次に，これらのさまざまなセンサーを整理し，それぞれのセンサーが感知できる物理量とその原理や性能について，簡単に説明していきたいと思います。

センサーのいろいろ

センサーを分類するときは，検出する物理量にしたがって分類するのが最もわかりやすいと思います。ここで説明するセンサーは光センサー(放射線センサー)，音波センサー(圧力センサー)，温度センサー，ガスセンサー，磁気センサー，位置センサーの6つです。

1） 光センサー

数あるセンサーのなかで，最もポピュラーで手軽に使えるのが光センサーでしょう。電化製品ではもう常識となっているリモコンはこの光センサーの応用そのものです。

現在使われている光センサーの大部分は，半導体を用いたフォトダイオードあるいはフォトトランジスタです。このうち，より構造の簡単なものがフォトダイオードで，これはP型半導体とN型半導体とを接合した構造をしていて，それぞれに端子が出ています（図1）。

フォトダイオードに光が当たったときにはダイオード内部のP型の部分には正の電荷が，N型の部分には負の電荷が生じ，それぞれの電荷が端子から電流として取り出せる仕組みになっています。そして，その電流の大きさ（内部に発生した電荷の量に相当する）が当たった光の強さにほぼ比例するので，光の強度を測定することができるのです。

フォトダイオードの分光感度はその材料によって決まりますが，半導体で最もよく使われているシリコンを材料とすると，波長で200～1100nmに感度があります。人間の可視光は400～700nmですから，ちょうど可視光全域をカバーすることができます。もちろん，人間の見えない近紫外光，近赤

図1 フォトダイオードの構造

光が入るとP型半導体内に正電荷，N型半導体内に負電荷が生じ，それを電極から電流として取り出す

外光にも感度があることになります。また，その感度の絶対値ですが，10^{-4}～10^{6}ルクスの極めて広い範囲をカバーします。これはおおよそ夜の星明かりから真夏の日差しまでに対応しています。さらに人間の目では感じることができないくらい微弱な光を検出するには，光電子増倍管という別の仕組みのセンサーを使いますが，詳しいことはここでは省略します。

桒野氏のガイガーカウンタの製作記事で詳しく解説されていた放射線も光とともに電磁波の一種なので，放射線センサーといっても今述べた光センサーと同じものです。しかし，放射線は可視光に比べて物質の透過力が強いので，放射能の吸収力の大きい物質を選びます。この点でシリコンよりもゲルマニウムやテルル化カドミウムという材料のほうがよいようです。

2） 音波センサー

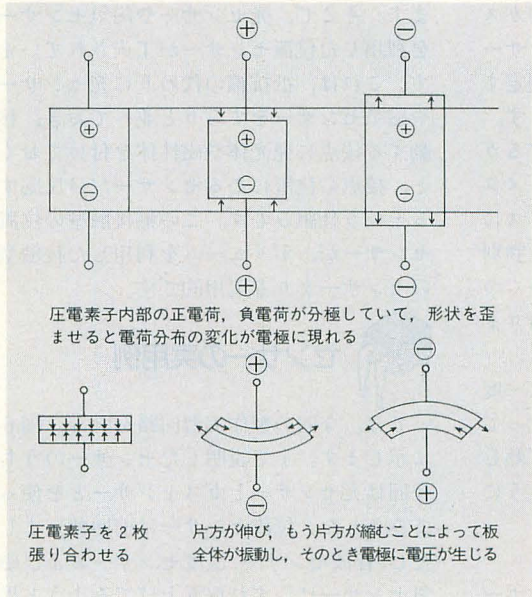
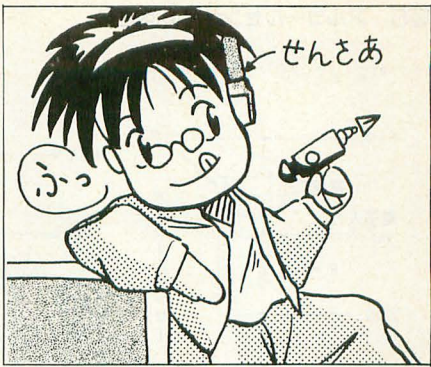
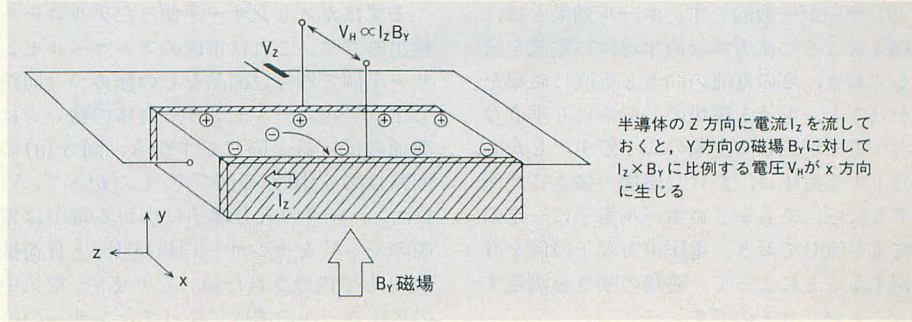
音波は空気を伝わってくる振動です。もし音波が物体に当たるとその振動エネルギーが物体に移って，物体も振動します。一番顕著な例が，ギター弦のチューニングをするときに，音叉をそばにおいて弦を弾くとその音が音叉に乗り移って音叉が振動し始める現象です。そこで，音波センサーはセンサー自身の力学的振動を電気信号に変える仕組みになっています。それには，圧電効果といって，歪みを受けると電圧を発生させる材料が用いられます(図2)。圧電素子にかかる歪みは音波の強さに比例し，したがって発生する電圧に比例しますので，音波センサーとして使えるわけです。

よく利用される音波センサーはいわゆるマイクです。このマイクの中にも圧電効果を示す素子が入っています。その構造は，セラミックまたは半導体の圧電素子を2枚張り合わせた板の両面から端子を取り出した形になっており，板全体が振動する際に生じる板両面の電位差を検出しています。

音波センサーはマイクのように人間の可聴領域にある音声を検出するだけでなく，オートフォーカスカメラの距離計や物体の速度・加速度計あるいは潜水艦用の水中探知器などにも使われています。これは，センサー自身から音波を発信し，物体に当たってはねかえり，再びセンサーに戻ってくるまでの時間を計測することにより，そのときの音波の速度から計算して物体までの距離を測定するのが基本になっています。このときは，可聴音だと耳障りなので，40kHz程度の超音波を使っていますが，センサーそのものの基本的な仕組みは変わりません。

3） 温度センサー

温度センサーも家庭の電化製品ではよく使われています。エアコンの室温調整には欠かせません。温度センサーの大部分も半導体を用いたものです。半導体に電圧をかけたときにその中を流れる電流の大きさは温度によって変わります。一般には，温度が高いほうが電流が流れにくく，温度の逆数に対して電流値は指数関数で変化します。

実用的には，トランジスタの電流－電圧特性を利用します。トランジスタにはエミッタ，コレクタ，ベースの3つの端子があり，エミッターコレクタ間に一定電圧をかけておくと，ベースからエミッタに流す電流値に比例してコレクタからエミッタへの電流値が変わる仕組みになっています（図3）。ところが，エミッターコレクタ間電圧とベースからエミッタへの電流の両方を一定値にしたとしても，周囲の温度によってコレクタからエミッタへの電流が変化するのです。そこで，このコレクターエミッタ間の電流を測定することによって，温度が測定できるのです。最近では，さらにこの電流を電圧にかえて，しかも出力電圧が直接摂氏温度に対応するようなICセンサーが普及しています。これによって，IC1個で特に外付け部品もなく，0～150℃の範囲が一発測定できるようになっています。

4） ガスセンサー

ガスセンサーは火災報知器などに使われていますし，工場などの安全管理にも欠かせません。また，あまり身近ではありたくないのですが，警察が飲酒運転の取り締まりを行うときのアルコールセンサーもガスセンサーの一種です。

ガスセンサーはセラミック，半導体ともに素子の表面にガスの分子が吸着したとき

図2 圧電素子の構造

圧電素子内部の正電荷，負電荷が分極していて，形状を歪ませると電荷分布の変化が電極に現れる

圧電素子を2枚張り合わせる

片方が伸び，もう片方が縮むことによって板全体が振動し，そのとき電極に電圧が生じる

図3 トランジスタの温度特性

コレクターエミッタ間に一定電圧V_{CC}をかけておき，ベース電流I_Bを流すと，コレクタ電流I_Cが流れる。
このとき，ベースーエミッタ間電圧をV_{BE}とすると，

$I_C=I_0e\times p\left[\frac{qV_{BE}}{kT}\right]$　q，k 定数

と温度Tの関数となる

図4 ホール素子の仕組み

半導体のZ方向に電流I_Zを流しておくと，Y方向の磁場B_Yに対して$I_Z \times B_Y$に比例する電圧V_Hがx方向に生じる

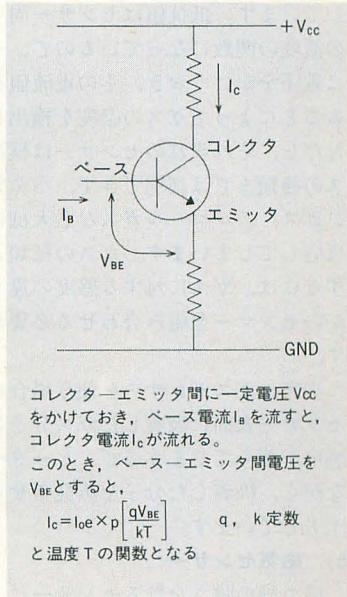



図5 アルコールセンサーの回路図

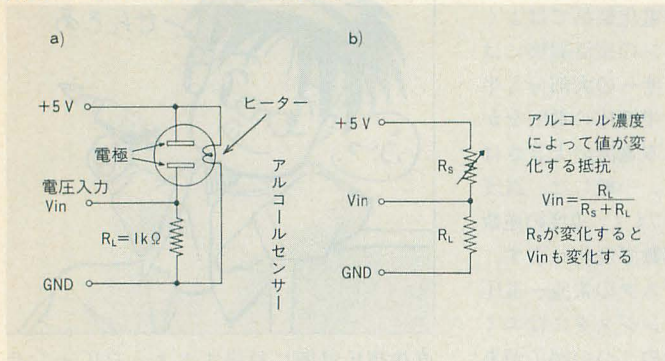



図6 光センサーの回路

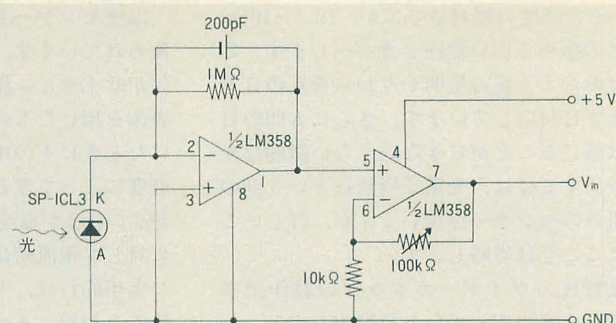
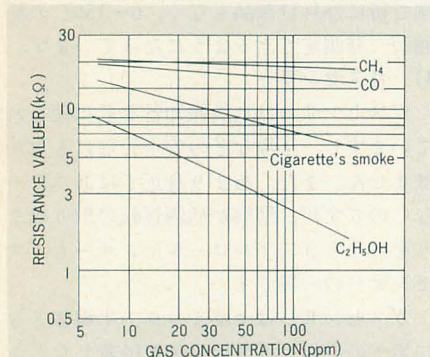



図7 Sensitivity characteristics

に電流に対する抵抗値が変わる性質を利用しています。抵抗値はセンサー周りのガスの濃度の関数になっているので，センサーに電圧をかけておき，その電流値を測定することによってガスの濃度を検出します。ただし，それぞれのセンサーは検出するガスの種類までは判別できず，水蒸気，メタンガス，アルコールガスなど大抵のガスに反応してしまいます。ガスの種類まで判別するには，ガスに対する感度の違ういくつかのセンサーを組み合わせる必要があります。

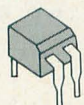
実際にガスセンサーを使う場合には一度センサー表面に吸着したガス分子によって感度が落ちてしまうので，ヒーターで熱しながら，吸着した分子を蒸発させるように工夫しています。

5) 磁気センサー

磁力線の強さを計るセンサーには，ホール効果というものを利用した半導体ホールセンサーが一般的です。ホール効果とは，図4のように直方体状の半導体に電流を流しておき，その電流の向きと垂直に磁場をかけると，電流と磁場のどちらにも垂直な面に電圧が生じる現象のことです。しかも，発生する電圧は，かけた磁場の強さに比例するので，あらかじめホール素子に一定の電流を流しておき，電圧出力端子の値を計測することによって，磁場の強さを測定することができるのです。

6) 位置センサー

位置センサーの最も簡単な例はボリュームです。前回のボリュームを使った簡易アナログジョイスティックはこの位置センサーのひとつの応用例といえます。これは，抵抗値が抵抗線の長さに比例することを利用したもので，詳しい仕組みは，前回説明したとおりです。

ところが，ボリュームを使った位置センサーには重大な欠点があります。それは，位置を検出するための接点が機械的に抵抗線と接触しているということです。機械的な接点は長時間使用していると必ず磨耗します。そこで，光センサーや磁気センサーを利用した位置センサーが工夫されています。これは，抵抗線の代わりに光センサーや磁気センサーをずらりと並べておき，移動する接点に発光体や磁性体を付けておくと，接点の位置にあるセンサーだけ反応するという仕組みです。この無接触型の位置センサーが，ボリュームを利用した接触型のセンサーよりも実用的です。

センサーの実用例

では，今回の製作実習回路を図5，図6に示します。上で説明したセンサーのうち，今回は光センサーとガスセンサーとを使ってみました。位置センサーは前回使いましたし，音波センサー，温度センサーおよび磁気センサーはいずれ取り上げてみようと思っています。

まずはガスセンサーを使ったアルコール検出器です。これは市販のアルコールセンサー1個で外付け部品なしの極めつけ回路です。アルコールセンサー自体の使い方は普通の抵抗器と同じですから，図5(a)の回路は図5(b)と等価です。したがって，A/Dコンバータの入力端子にかかる電圧は電源の+5Vをセンサーの抵抗 R_S と負荷抵抗 R_L とで内分された値になります。空気中のアルコールの濃度によってセンサーの抵抗値が変わるので，それに応じて出力電圧が変わる仕組みになっています。参考までに，図7にアルコール濃度とセンサー抵抗との関係を示したグラフを挙げておきます。

次はフォトダイオードを使った光センサーです。図6を見てのとおり，LM358というIC1個を付け足しています。このLM358はオペアンプというアナログ信号の増幅器2個からなっています。このオペアンプを2段組み合わせた回路は，フォトダイオードが光を受けて出力した電流を電圧に変え，しかもそのままの電圧ではA/Dコンバータで測定するには小さすぎるので，測定可能な電圧値にまで増幅するものです。1段目は電流増幅器，2段目は非反転の可変増幅器というものになっています。

お馴染みのTTL-ICは基本的なデジタル回路をパッケージに納め，世界的な統一規格にしたがって通し番号が付けられていますが，このオペアンプは基本的なアナログ回路をパッケージ化したものです。使い方もTTL-ICと同様に規格にしたがって端子間をつなぐだけの簡単な作業ですみます。とはいえ，アナログ回路はデジタル回路よりもさらに皆さんの馴染みのうすいものではないでしょうか。

というわけで，来月はたっぷりと1回分を割いてオペアンプによるアナログ回路入門を解説したいと思います。そのときに上で述べた「電流増幅器」，「非反転増幅器」などの言葉の意味にも触れるつもりです。さらに，フォトダイオードの使い方も一緒に説明したいと思いますので，光センサー回路の詳細は次回までお待ちください。

*

以上，今回はセンサー理論編として，さまざまなセンサーの種類とその仕組みについてひと通り解説してみました。もちろん文章で説明するだけでは理解しにくいと思いますが，来月はもう1回理論編で我慢してもらうことにして，再来月には製作実習に移りたいと思いますので，お楽しみに。

●特集 急接近! SX-WINDOW

SX-WINDOW

SX-WINDOWとはなにか。それは誰のためにあるのか。ひとつの答えは「アプリケーションの実行環境であり，ユーザーにとってわかりやすい操作環境を提供するシステムである」ということだ。だから本来，エンドユーザーはウィンドウ上でプログラムが走るというのがどういうことかなど考える必要はない。しかし，あえて本誌の読者にはパソコンの未来のために考えてほしい。現時点でSX-WINDOW上のアプリケーションはまだ市販ソフトのレベルでは存在しない。実際のところ，ソフトメーカーに対してもまだ資料やツール類のサポートが始まったばかりの段階だ。だからそのような時期に，付録ディスクによって膨大な資料を読者の全員に配布できたのは、幸運なことである。確かにわかりやすいものではないが挑戦してほしい。X68000の，あるいはパソコンの新しい環境づくりに私たちユーザー自身が積極的に参加できる機会を得られたのだから。

¥WORKS¥CONTENTS

CONTENTS

SX-WINDOWへの心構え

美しい環境を目指して

Ogikubo Kei 荻窪　圭

X68000の新しい操作環境として登場したSX-WINDOWですが，ようやくアプリケーションを開発できる環境が整いつつあります。そこで，ウィンドウ環境のあるべき姿を考えつつ，私たちも心構えを新たにすることにしましょう。

きしむマウス
うなるハードディスク
たゆたうメモリ
SX-WINDOWがやってきた
ヤア！ ヤア！ ヤア！

てなわけで，机に肱をつき，掌に顎を乗せ，少し斜めにCRTを眺めながら，右手でマウスをなでる。おおあくびひとつと，適当なクリック。ずりずりとアイコンを動かし，惚けた顔でアプリケーションを動かす。それがSX-WINDOWだと思ってはいないか。

確かに間違いはないかもしれない。ただし，それはエンドユーザーだけに許される姿だ。エンドユーザーが楽をするにはプログラマが苦労をし，プログラマが楽をするにはシステムの開発者かエンドユーザーが苦労する。ところが，エンドユーザーは楽をしているし，システムは出来上がっている。

そう，掌に顎を乗せてあくびをしている場合ではないのだ。

さもありなん。エンドユーザーに徹するのもよいが，SX-WINDOWの資料も（最低限ではあるが）出揃ったいま，SX-WINDOWワールド形成に向かっての心構えについて考えてみたい。ちゃんと百八つの煩悩を洗い，一年の圭を，じゃなかった計を元旦に立てるのだ。

VS.Xはなんだったのか

ビジュアルシェルがいた。アイコンをダブルクリックすると，プログラムが実行された。やったあ。

さあ，使うぞ。あれ？　マウスが使えない。どうしてだ。せっかくのマウスじゃないか。よし，マウス対応のアプリケーションを立ち上げるぞ。あれ？　スクロールバーの形が違う。ファイルがアイコンで表示されない。右ボタンが使えない。クリップボードや電卓や，ソフトウェアキーボードはどこへ行った？

どこやらのアプリケーションを責めるつもりはない。よくよく考えてみれば当たり前だったのだ。なぜなら，ビジュアルシェルもひとつのアプリケーションだったから。

しかし，われわれは期待していた。同じPRO-68Kシリーズならきっと操作性も一緒で，データのやりとりも簡単にできるよね。……で，できない。

終了するとビジュアルシェルに戻る。じゃあ，ビジュアルシェルってなにをやってるんだ。あのウィンドウはファイルアイコンを表示する以外に能はなかったのか。ファイルの中身ひとつ見る機能がないぞ。

あーあ，こんなことなら，コマンドシェルにしよっと。

そういう流れを多くの人がたどったに違いない。少なくとも私はそうだった。ビジュアルシェルからなにかを実行することはできても，ビジュアルシェルの上で実行することはできなかったのだ。

SX-WINDOW。プログラムを起動しても消えてしまわず，どのアプリケーションも同じような画面，操作性を貫くことができ，いつだって電卓が使える。SX-WINDOWの上でプログラムを実行することができるようになったのだ（というよりSX-WINDOWの上で動くプログラムを書けるようになったのだ）。

せっかくだ。せっかくだから，ビジュアルシェルの轍を踏みたくない。まだろくなアプリケーションもないSX-WINDOWを秩序のない乱れた世界にしたくはないだろう。

だから，次の行からそういう話が始まる。

1）とりあえず，とっついてみるべし

とりつかねば何も始まらない。今までの環境に慣れた体・心・頭に新しいSX-WINDOWはうっとうしいに違いない。しかし，慣れたものにいつまでもしがみついていたら，人間は変化を止めてしまう。変わることを恐れるようになったら，もう残りの人生は余生といわれてもしかたがない。

食わず嫌いはよくないのだ。

2）最初は面倒だと観念すべし

ウィンドウシステムでプログラムを書くというのはおしなべて最初は面倒である。新しい規則，新しい概念，新しい用語。今まではいきなり自分のやりたいことだけを記述すればよかったのに，初期化は面倒臭いし，イベントがどうしたこうした，タスクマンの相手から，ウィンドウマンやコントロールマンうんぬん。目的のアルゴリズム以外に書くことが多すぎる気がするだろう。しかし，それらはみな慣れてしまえばワンパターンの朝礼のようなもの。さまざまな恩恵をこうむることを思えば，朝礼なんかよりずっと花も実もある。面倒な手続きは手続きのための手続きではなく，自由と秩序のための手続きなのだ。

3）曲芸で客を沸かすのは10年早い

よくいるんだ。うんちゃらマンをいちいち呼び出すより，自分でアセンブラで書いてしまったほうが早いと思うやつが。それはそれで楽しいだろうが，まだちょっと早すぎる。環境が整わないうちに環境を混乱させるアクロバットはただ邪魔なだけだ。まずは同じルールのもと，秩序を作るところから始める必要がある。

4）デザイナーたれ

SX-WINDOW上に展開するすべてのプログラムは美学を必要とする。男のロマン

といってもいい。せっかく4階調モノクロで，影文字やカッコいいアイコンがあるのだ。そこにダサいやつらが紛れ込むのは美しくない。たとえば，文字は12ドット影文字だ。装飾は渋く。ほこりまみれの単純アメリカンより伝統の屈折した美学のヨーロピアン。ちょっと斜めからカッコつけてみる姿勢がよく似合う。ひたすらカッコよく，なおかつ，自分の世界に溺れないようなセンスが重要だ。

5）私だけの操作環境を作り出すな

せっかくのウィンドウシステムである。操作環境は標準アプリケーションの作ったものを（とりあえず）踏襲すべし。あまりにも新感覚なスクロールバーなど疑問符を背中に背負ったようなものは別としてだ。左クリックでポップアップメニューが出現したり，ウィンドウクローズボタンが右下にあったり，トリプルクリックなどを駆使したりはしないように。さらに，プルダウンメニューしかないとか，ポップアップのカット&ペーストが使えないのもいやだ。

また，せっかく，ファイル（オブジェクト）をアプリケーションウィンドウにドラッグして引き渡すというオペレーションがあるのだから，そういういいところは採用する。間違っても自分のプログラムでファイル一覧をし，そこからファイルを選択させたりしないように。

クリップボードも，使えるかどうかは別にして，せっかくあるものだから，クリップボードを介したテキストのカット&ペーストは文字入力可能なすべての局面で採用してほしい。

くれぐれも，アプリケーションひとつで「私だけの世界を作らない」ことが重要だ。アプリケーションがみんなで我を張り始めると，ウィンドウシステムの意味がひとつなくなってしまう。

6）恥ずかしい真似はするな!

恥ずかしい真似というのは，たとえば，「さ○りん」なんて恥ずかしい名前をウィンドウに入れるとか，そういったことだ。これは恥ずかしくてダサいことなのだ。

SX-WINDOWは仲間内で閉じた世界ではなく，どのアプリケーションにも均等に開かれた世界である。趣味のプログラムならともかく，少しでもSX-WINDOWという環境を作り出す一端を担うのなら，恥ずかしい真似はすべきではない。子供っぽい自己顕示は笑われるだけだ。みこしを担ぐ必要もない。

7）閉じた世界でワイワイはやめたい

SX-WINDOWはとても狭い。狭いのはCRTなのだが，その狭い空間で何をするか。ただ絵を動かし楽しむより，そこを自分の環境として，生活や作業の一部を担わせたい。

今しか笑えないいじけた娯楽より，自分を磨く明日の娯楽だ。スケジューラでもタイムテーブルでもメモ帳でもデータベースでもドローイングでもいいが，SXシェルというのは作品発表の場であると同時に，エンドユーザーがなんらかの作業をする場でもあるのだ。足元を固めないと，ただ楽しいだけで終わってしまう。すべてを終わらせるのは膠着した心だ。

8）カッコつけるだけでは意味はない

で，デザインを気にして，操作性を統一して，悪さしないで，とそこまでいうからには，それなりのメリットがあるわけである。

たとえば，複数のアプリケーションを開いて，互いにデータをやりとりする。左に通信ソフトを開いて，右にワープロ。通信のログから必要なところをワープロにリアルタイムでカット&ペースト。また，ワープロで作った文書を通信ソフトにペーストとすると，ほら簡単にアップロード。

住所録。電話番号部分を通信ソフトの電話ボタンに渡してやると，オートダイアル。

いままでの環境では同じソフトハウスが全部整合性を考えて作らないとできなかったものが，SX-WINDOWなら誰のプログラムでも（ちゃんとつくれば）こういったデータのいったりきたりが簡単にできるのだ。

まだあるぞ。グラフィックを張り付けられる15パズルと，グラフィックエディタがあるとしよう。片や絵を作成しながら，片や絵をファイルアイコンをパズルのウィンドウにドラッグする。すると，1～15の数字の代わりにその絵が埋め込まれるとか。

一度に同じアプリケーションをいくつも起動できるのもおいしいかもしれない。ワープロを2つでもいいし，スプレッドシートを2つでも3つでもいい。

私が欲しいのは，文書整形印刷ツールだな。テキストファイルのアイコンをドラッグしてやって，ちょいちょいと設定してやると印刷してくれる。こうすると，書く道具と印刷する道具を分けることができる。

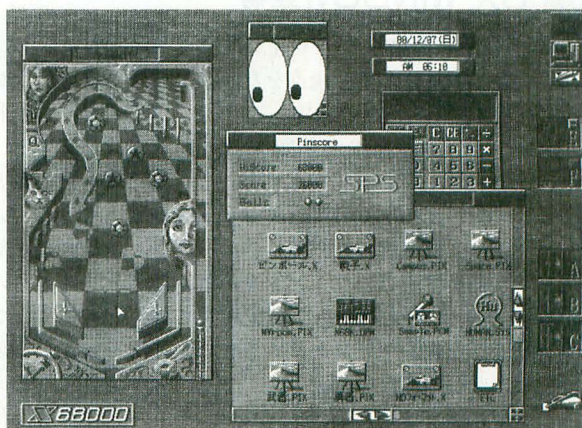
それに，ある作業をしている最中に，ちょっと別のことをしたくなったら，いままでは一度クローズするかチャイルドプロセスを起動するかしかなかっのだが，SX-WINDOWならそいつのアイコンをダブルクリックしてウィンドウをまた開けばすむのだ。

ワープロで仕事をしているとき，ああ，疲れたな，と立ち上げてはあるのだが途中でポーズにしてあるゲームのウィンドウをアクティブにして，ゲームやって，ゲームも疲れたな，とまたワープロに戻って仕事ってのも簡単だ。勉強するふりをして遊ぶのには欠かせない。いままでは，疲れたなといっても[OPT.1]+[.]でテレビを見るくらいしかできなかったからね。

ほら，こう考えると，画面をいっぱいに使って作業しているとき，ダサいやつが混じっていると情けないでしょ。こういうことができるというのに，ワープロが私だけの世界を作って，変なファイル操作をしたり，通信ソフトだけが左ボタンでポップアップメニューだったりしたら困るでしょ。

そういうことなのだ。

＊

SX-WINDOWがウィンドウシステムとしてどれだけ使い物になるか，果たしてMacやMS-WINDOWSのように使えるアプリケーションは現れるのか。まだ未知数の部分は大きい。われわれユーザーがやることは，手探りながらでもSX-WINDOWの可能性を追求していくことである。どんな可能性でも見つかれば楽しいし，そこから何かが生まれれば面白い。SX-WINDOWというのは気が向いたときに使うツールではなく，常時触る環境なのだ。その気になればなんだってできる。なんだって。

入門のための基礎知識

SX姫と15人の小人たち

Yoshida Kouichi　吉田 幸一

SX-WINDOWを支えるもの，いや，その実体は15個に分けられたマネージャたちです。マルチウィンドウ，マルチタスクに必要な処理をそれぞれ分担しています。まず，ウィンドウの基礎知識とこれらのマネージャの概要を見てみましょう。

そういえば，SX-WINDOWが発売されてから何カ月にもなる。その間，猪木はイラクで吠えるわ，亜久里は3位になるわ，サッチャーは辞任するわ，即位の礼は行われるわとまあいろいろ世間は変わった。

SX-WINDOWを買ってしまった皆さん，どうしているだろうか。私が2回ほど紹介したあと，すっかり音沙汰がなくて不安だったのではないだろうか。不安だったのは私も同じである。

もしかしたら，SX-WINDOWを高級なビジュアルシェル(VS.X)だと思っている人もいるかもしれない。SX-WINDOWを使えば夢を超えるような環境が出現すると信じている人もいるかもしれない。SX-WINDOWなんてクソだぜ，って思っている人もいるかもしれない。まあ，なんだっていいけど。MS-WINDOWSだってVer. 2.11まではクソだったのだがVer 3.00になって少しはまともになったように，いまのSX-WINDOWがクソだと思っても，明日のSX-WINDOWは脳天気かもしれないのでそう簡単に投げないように。投げるのはいいけどな，拾ってこられるから。捨ててはいけないぞ。ゴミ収集車に持って行かれたら復活しないから。

というわけで，SX-WINDOWの歴史は今日から始まる。昨日までは紀元前だったのだ。

SX-WINDOW登場

SX-WINDOWのようなウィンドウシステムを語る際，見る者の視点をなおざりにしてはならない。

エンドユーザーの視点と開発者の視点だ。いままで，ウィンドウシステムといえばエンドユーザーからの視点ばかりが強調されてきた。開発者にとってみれば「マウスをどう動かすとこのウィンドウが動くのか」ってことより「どのように記述すればウィンドウを動かすプログラムを書けるのか」のほうが重要なのだ。

今日から歴史は始まる，としたのは，やっと開発者の視点でもSX-WINDOWを見られるようになったからだ。つまり，SX-WINDOWの中身や規則が公開されたのである。それが付録に圧縮されて詰まっている膨大な技術資料だ。私が書くのは，そのさわり。エンドユーザーにとって重要な，目に見えない部分でのSX-WINDOWがなんとなくわかるような話。

さて，OSというものがある。パソコンの世界ではデータや文字コードの共通化やコマンドのサポートくらいしかありがたみがないように思われているが，本来は違う。大きなコンピュータの世界でのOSはもっと権力を持っており，OSを通さずにはどんな些細なプログラムでさえ動くことができないのだ。ユーザーのできるすべてのことがOSを通してなされる。そうしないと，コンピュータがちゃんと動かない。OSの目的というのはそういうコンピュータ資源の管理であり，同じOSなら同じプログラムが動くというのはその次の問題だ。

パソコンがそうでないのは性格上の問題で，ちゃんとしたOSを載せるほど管理する必要のある資源もなかったし，そういうOSを載せられるほどメモリもなかった。それに，一度にひとりしか使わず，一度にひとつのプログラムしか実行されない環境だから，個々のプログラムがなにしようと，立つ鳥あとを濁さなければ構わなかったのだ。

それが，ウィンドウシステムとなるとちゃんとOSしなければならなくなる。一度にいくつかのプログラムが立ち上がったりし，ハードウェアをいろんなプログラムで共有する。そうすると，悪さするプログラムがあると，ほかのプログラムに影響が出てくる。いままではプログラムひとつにつき画面全部を好きにできたのが，ウィンドウシステムになると，自分のウィンドウしか好きにできない。こういったさまざまな制約が必要となってくる。

とりあえず，制約が増えて自由に書けなくなった代わりに，データさえ与えればいろいろやってくれる小人がたくさんいる，と思えばいい。ただ，小人を動かすための手続きや言葉，動かし方を覚えるのが面倒なだけで，わかってしまえばそれほどでもない（という話だ）。

このようにウィンドウシステムが複雑な処理を行い，プログラマがそれに対応するプログラムを書くと，皆さんが共存共栄するわけである。万歳（私は海部かって）。Macを嫌いだという人はよく規則でがんじがらめだからという。それもひとつの考えだがね。

ウィンドウシステムとは

ふつうウィンドウシステムというと，使うことがメインであり，自分でプログラミングして使おうなどとは考えないものだ。UNIX上のX-WINDOWならともかく，たとえば，「MS-WINDOWS用アプリケーションをプログラミングしよう」というのは，やむにやまれぬシステムエンジニアくらいのもので，自発的にそうしようとするのはそうとう奇特な人と思われている。

にもかかわらず，SX-WINDOWの開発資料を心待ちにしていた人は多いはずだ。しばらく前のシャープの広告を見ると「こだわり続けてきたある執着がまさに帰結しようとしています」と，ある。開発者もこだわっていたそうだが，ユーザーはもっとこだわっていたのだ。

さて，SX-WINDOWでの開発言語はC言語またはアセンブラとなる。

特に今回はアセンブラでのアプリケーション作成を主眼とした特集だ。「どうやったらSX-WINDOW上で動作するプログラムを作成できるか？」について解説するのだから，アセンブラでのプログラム作成能力は基本的な前提だ。

昨年（1990年）6月号のアンケート発表ではアセンブラプログラマが23%とCプログラマの21%をわずかに凌駕する。8ビット機ならともかく，16ビット機以上でこんなにアセンブラ野郎がいるマシンは空

前絶後といっていい。

そんなアセンブラがバリバリに使える人には必要ないかもしれないが，一応ウィンドウシステムでの概念とディスク内のドキュメントを読むための用語の整理を行っておくことにする。

●ウィンドウ

画面のなかにある別の画面。ひとつの画面内にたくさんの画面を開いて，そこで別々のことをするのがマルチウィンドウだ。WP.XやmicroEMACSの画面分割はタイリングマルチウィンドウ，VS.XやSX-WINDOWはオーバーラップマルチウィンドウ。流行りのオーバーラップではビットマップディスプレイ，ポインティングデバイスが揃いで出てくることが多い。ビットマップとはちまたでいうグラフィック画面（ただし色は関係なし）のこと，ポインティングデバイスは画面上の任意の点を指示するための装置のうち，カーソルキーよりは使い勝手のよいものを指す。たいていはマウスのことだ。

●アクティブウィンドウ

ウィンドウがいっぱいあっても，マルチタスクしていても，ユーザーの操作を直接受け付けるのは基本的にひとつのウィンドウだけだ。これがアクティブウィンドウ。概念的にはいちばん上に重なったウィンドウ，SXシェルではなんとなく枠が違うやつなので見分けることができる。

シングルタスクのMacintoshなら「アクティブ」はわかりやすいが，SX-WINDOWはマルチタスクだからタスクがアクティブでもアクティブウィンドウとは限らない。暁子さんはウィンドウの下でも動くでしょ。

●アクティベート処理

アクティブウィンドウにすること。通常，そのウィンドウがあらゆるウィンドウの上にくるように画面が書き直される。

●マルチタスク

SX-WINDOWの暁子.Xが例に挙げられるが，実際には画面に出ているすべてのウィンドウでマルチタスクできる。動かないように見えるのはそのようにプログラムしてあるだけの話。

リレーの第1走者が審判員からバトンをもらう。所定の区間を走って第2走者にバトンを渡す。ただしアンカーはいない。みんな走り終わるとまた第1走者から回り始める。これが十分に高速なら，みんなバトンを持っているように見える，はず。SX-WINDOWではなにかイベントがあれば一斉に回り始める。たまに回んない奴も出てくるが，なにも指定しなければマルチタスクだ。

一定時間でバトンを渡すように細工したのがよくあるタイムシェアリングシステムというやつで，SX-WINDOWとは関係ない。時計もなしに一定時間で走れというのは無理だし，スェーデンリレーかもしれない。SX-WINDOWでは，誰でも好きなだけ走っていい。が，ひとりがあまり走りすぎるとマルチタスクに見えないので，しかるべく配慮すること。

●ヒープ

SXシステムではフリーエリアをヒープ領域とスタック領域に分けて管理する。プログラムやデータが入るのはヒープ領域だ。ヒープってのは聞き慣れない人にはまったく聞き慣れない言葉だが，元の意味は塊とかどっさりとかそんなものだ。気にしなくてもいい。情報処理業界ではちゃんとそれなりの意味はあるのだが，気にすることもないだろう。ダイナミックにメモリを割り付けるための領域を指す言葉だ。

当たり前だが，どのプログラムがメモリのどれだけを確保しているか，ちゃんと管理しなければならない。その単位をブロックという。ヒープ領域をブロックごとに管理しているわけだね。ブロックにはリロケータブル・ブロックとアンリロケータブル・ブロックがあって，リロケータブルというのはメモリ空間のどこに置いてもいいよ，っていう意味。こちらが基本。僕はここしかいやだ！　という駄々をこねるやつがたくさんいたら困るのだ。このメモリ・マネージャさんのおかげで，プログラマの考えることが少し減る。任せてしまえばいいから。

●クリップボード

基本的にウィンドウシステムでは各ウィンドウ間でデータをやり取りするための方法が用意されている。そのうちのひとつがMacintoshで使われているクリップボードだ。SXシェルにも用意されているが，Macintoshではテキストデータだけでなくビットマップデータも保存できるので便利。

メモリ上のデータフォーマットが統一されているならデータのポインタをこっそり教えてあげたほうが早いかもしれない。

●リソース

リソースというのは資源という意味である。が，この業界で資源というとあまりに

ウィンドウの構造

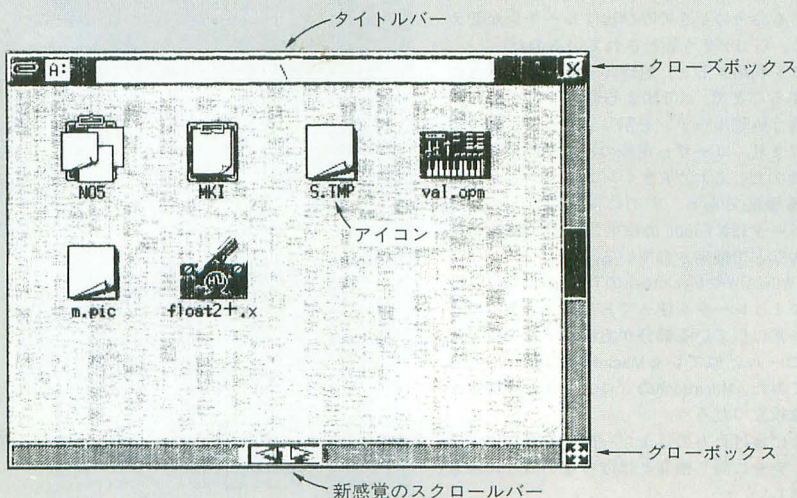
SX-WINDOWでの標準的なウィンドウの各部を見てみよう。まず，いちばん上のタイトルなどが書いてあるところは，ずばり「タイトルバー」という。タイトルバーの上で右端にある×印は「クローズボックス」。その名のとおり，ウィンドウを閉じるためのものだ。

ウィンドウの右端と下側についている部分は「新感覚のスクロールバー」でウィンドウをスクロールさせるためのもの。どう動くのかはよくわからない。

右端隅の矢印つきの四角形は「グローボックス」ウィンドウの大きさを変えるときに使う。

ウィンドウの大きさが変わらないとわかっているときはタイトルバーの部分と本体だけのウィンドウもある。

さらにタイトルバーもなく，四角形がどんと出るものをダイアログという。決まって画面の真ん中に出てきて移動できない。「確認」とか「取消」というスイッチのついたやつだ。たいていスイッチが並んでいて，対話的にさまざまな設定を行うものだそうだ。出ているあいだ処理が止まるのでハードコピーも取れなかった。困ったもんだ。

タイトルバー
クローズボックス
A:
NO5
MKI
S.TMP
val.opm
アイコン
m.pic
float2+.x
グローボックス
新感覚のスクロールバー

も意味が広い。ハードウェアすべてから，ときには人間までコンピュータ資源のひとつとして扱うことになる。それではあまりにもあいまいなので，資源とはいわずにリソースという。

リソースというのは，ソフトウェア資源の一種。さらに狭義に，アプリケーションが扱うデータだと思って構わない。さらに正確にいうと，ここでいうリソースとは，固有名詞である。リソースという名前の形式のデータがあって，リソース・ファイルというファイルに入っているのだ。その中身は音であり，フォントデータであり，グラフィックデータであり，メッセージであったりする。アプリケーション・プログラムとリソースを区別することにより，プログラムのデータ部分が独立するのである。

マネージャがたくさん

SX-WINDOWはウィンドウシステムである。ウィンドウシステムといってもSX-WINDOW全部をひっくるめてウィンドウシステムといっているのであって，ひっくるめてしまうとよくわからない。

SX-WINDOWの基礎として，SX-WINDOWを2つに分けて考えよう。

ひとつは実際に稼働するシステムの部分である。シェルであるSXWIN.Xと，デバイスドライバであるFSX.Xがそうだ。COMMAND.Xをコマンドシェルというのと同様，SXWIN.XはSXシェルという。FSX.Xが実際にいろいろ資源を管理したりコントロールしたりする。この2つが働いて，目に見えるSX-WINDOWというものができあがる。エンドユーザーはこのシェルとしての顔しか目に触れないわけだ。

ちなみに，シェルというのは昭和シェル石油のシェルと同じ。貝の殻のことであり，殻のようにOSを包んでいるのでシェルと呼ばれるようになった（らしい）。

もうひとつは開発支援としてのSX-WINDOWシステムである。これはCやアセンブラのライブラリとして提供され，プログラムする人は自分で余計な世話はせず，みんなこれらに任せなければならない。下手に自分でやると，行儀が悪いといわれてしまう。

さて，こういってしまうと単純だが，SX-WINDOWはウィンドウシステムとしてさまざまな資源を管理するため，動作するのも大変である。コマンド環境のときは最低限の入出力とBIOSを提供しておいて，あとはアプリケーションに任せていたので，システムは少ないメモリで動作した。ウィンドウ環境ともなると，その環境で使われるすべてについて管理せねばならないので，システム自身がたくさんのメモリを必要とし，さらに「グラフィカル」なユーザーインタフェイス用にもメモリを使う，そして要2Mバイトとなってしまう。ご苦労さんである。

そういった管理をするために，SX-WINDOWはたくさんのマネージャを持っている。SXシェルはあくまでもシェルであり，働くのはこのマネージャたちなのだ。プログラマがプログラム中でSX-WINDOW用ルーチンをコールすると，コールされたルーチンたちが実行時にマネージャをこき使うのだ。マネージャはたくさんいて，それぞれが自分の大事な仕事を抱えている。

MacintoshとSX-WINDOWの人にはいえない怪しい関係

「SX-WINDOWのシステムコール（SXコール）はAラインエミュレータによって行われる」

これは，X68000のCPUであるMC68000の例外処理動作を利用したものだ。通常，CPUはメモリ上からプログラムを読み込んで解釈実行する。このとき，CPUの仕様に定められていない命令を実行しようとすると，CPUはプログラムが暴走した危険があるとして，あらかじめ定められたエラー処理プログラムを呼び出す。

68000では命令コードを16進数で表したときFとAで始まる命令を持っていない。ほかにも抜けている命令があるのだが，これらは特に未実装命令と呼ばれている（それ以外は不当命令）。これらの未実装命令はAで始まる命令とFで始まる命令のときで呼び出すルーチンが変えてある。CPUがそう設計されているのだ。

これを利用すると，処理ルーチン側でさらに分岐することで，Aで始まる命令それぞれに対して違う処理ルーチンを割り当てることができる。つまり，ユーザー定義の命令が簡単に追加できるのだ。これがAラインエミュレータと呼ばれる機能である。すでに同様のFラインエミュレータはX68000のDOSコール，数値演算コールなどで使用されている。

SX-WINDOWやMacintoshのToolboxはこのAラインエミュレータを使っており，システムコールも非常に似ている部分がある。そこで編集室でSXコールと似ているMacintoshのコール名を調べてみた。Macintoshのプログラムを移植する際には役立つだろう。

ここに挙げたものはまだ一部にすぎない。さらに，完全に同じ機能とは限らないことに注意してほしい。

A-Emu	SX-CALL	MAC-CALL
$A29A	__CMCheck	TrackControl
$A28A	__CMDispose	DisposeControl
$A28E	__CMDraw	DrawControls
$A28F	__CMDrawOne	Draw1Control
$A29E	__CMDraws	UpdtControls
$A299	__CMFind	FindControl
$A28C	__CMHide	HideControl
$A28B	__CMKill	KillControl
$A295	__CMMaxGet	GetCtlMax
$A294	__CMMaxSet	SetCtlMax
$A293	__CMMinGet	GetCtlMin
$A292	__CMMinSet	SetCtlMin
$A296	__CMMove	MoveControl
$A289	__CMOpen	NewControl
$A29B	__CMRefer	GetNewControl
$A298	__CMShine	HiliteControl
$A28D	__CMShow	ShowControl
$A297	__CMSize	SizeControl
$A29C	__CMTitleGet	GetCTitle
$A29F	__CMTitleSet	SetCTitle
$A291	__CMValueGet	GetCtlValue
$A290	__CMValueSet	SetCtlValue
$A2CF	__DIGet	GetDItem
$A2D8	__DIHide	HideDItem
$A2D0	__DISet	SetDItem
$A2D9	__DIShow	ShowDItem
$A2D1	__DITGet	GetIText
$A2D3	__DITSelect	SelIText
$A2D2	__DITSet	SetIText
$A2D6	__DIUpdate	UpdtDialog
$A2D7	__DMBeep	TinkerBell
$A2C5	__DMClose	CloseDialog
$A2C7	__DMControl	ModalDialog
$A2C6	__DMDispose	DisposeDialog
$A2C8	__DMDraw	DrawDialog
$A2F6	__DMError	ErrDialog
$A2C2	__DMFontSet	SetDAFont
$A2C0	__DMInit	InitDialogs
$A2C3	__DMOpen	NewDialog
$A2C4	__DMRefer	GetNewDialog
$A2F8	__DMWaitClose	WaitDialogOff
$A2F7	__DMWaitOpen	WaitDialogOn
$A2F9	__DMWaitWhile	WaitDialogAnim
$A0B1	__EMBlinkGet	GetCaretTime
$A0B6	__EMBlinkSet	SetCaretTime
$A0B2	__EMClean	FlushEvents
$A0B0	__EMDClickGet	GetDblTime
$A0B5	__EMDClickSet	SetDblTime
$A0B8	__EMDeCross	DeCrossing
$A0B7	__EMEnCross	EnCrossing
$A0A5	__EMGet	GetNextEvent
$A0A2	__EMInit	InitEvent
$A0AE	__EMKMapGet	GetKeys
$A0A8	__EMLBttn	Button
$A0AA	__EMLStill	StillDown
$A0AC	__EMLWait	WaitMouseUp
$A0A7	__EMMSLoc	GetMouse
$A0B3	__EMMaskSet	SetEventMask
$A0A9	__EMRBttn	Button
$A0AB	__EMRStill	StillDown
$A0AD	__EMRWait	WaitMouseUp
$A0A6	__EMScan	EventAvail
$A0A4	__EMSet	PostEvent
$A0AF	__EMSysTime	TickCount
$A0A3	__EMTini	TiniEM
$A266	__MNInit	InitMenus
$A267	__MNRefer	GetMenu
$A268	__MNSelect	MenuSelect
$A20F	__WMActive	FrontWindow
$A218	__WMAddRect	InvalRect
$A219	__WMAddRgn	InvalRgn
$A200	__WMCarry	SendBehind
$A20A	__WMCheckBox	TrackBox
$A20B	__WMCheckCBox	TrackGoAway
$A1FB	__WMClose	CloseWindow
$A1FC	__WMDispose	DisposeWindow
$A205	__WMDrag	DragWindow
$A225	__WMDragRgn	DragGrayRgn
$A20C	__WMDrawGBox	DrawGrowIcon
$A1FD	__WMFind	FindWindow
$A21C	__WMGScriptSet	SetWindowPic
$A224	__WMGetDTGS	GetDeskTopPic
$A204	__WMGrow	GrowWindow
$A208	__WMHide	HideWindow
$A1F8	__WMInit	InitWindow
$A202	__WMMove	MoveWindow
$A1F9	__WMOpen	NewWindow
$A1FA	__WMRefer	GetNewWindow
$A1FE	__WMSelect	SelectWindow
$A201	__WMShine	HiliteWindow
$A207	__WMShow	ShowWindow
$A209	__WMShowHide	ShowHide
$A203	__WMSize	SizeWindow
$A21A	__WMSubRect	ValidRect
$A21B	__WMSubRgn	ValidRgn
$A221	__WMTIDGet	GetWRefCon
$A220	__WMTIDSet	SetWRefCon
$A21F	__WMTitleGet	GetWTitle
$A21E	__WMTitleSet	SetWTitle
$A20D	__WMUpdate	BeginUpdate
$A20E	__WMUpdtOver	EndUpdate
$A206	__WMZoom	ZoomWindow

彼らはどこにいるか。FSX.Xの中だ。

マネージャというのは管理者という意味。SX-WINDOWの持っているマネージャは次のとおりだ。

なお，マネージャの名前が「～マン」となっていて，少し恥ずかしい気もするが，メモリマンと一般的な意味でのメモリマネージャを区別する必要がある場合などを考えると，まあ，しかたないと納得しておこう。そのうち，もっと凄い「グラフマンタロウ」とか「ダイアログマンジョーニアス」とか出てくるかもしれないし。

MEMORY：メモリマン

メモリマネージャのこと。メモリ全部の管理をする。新しくアプリケーションが起動するとメモリを確保し，クローズすると開放する。そんな大事な仕事だ。メモリマンは日夜陰で働いているのだ。

EXCEPTION：エクセプションマン

エクセプションマネージャのこと。エクセプションというのは割り込み。V-DISP（垂直帰線）割り込みをサポートする。アプリケーションで使うことはあまりないだろう。縁の下グループの一員。

MOUSE：マウスマン

マウスマネージャ。マウスの座標計算をしたりする。マウスカーソルを表示するときにはアニメーションマンを呼んでやってもらう。マウスがクリックされると，イベントマンを呼んで後始末をしてもらう。それが仕事の彼である。

ANIMATION：アニメーションマン

アニメーションマネージャ。アニメーションといってもCGAしてくれるわけではないので誤解しないように。マウスカーソルのアニメーションを担当するやつだ。マウスカーソルのアニメーションはV-DISP割り込みを使うので，エクセプションマンと仲好しである。

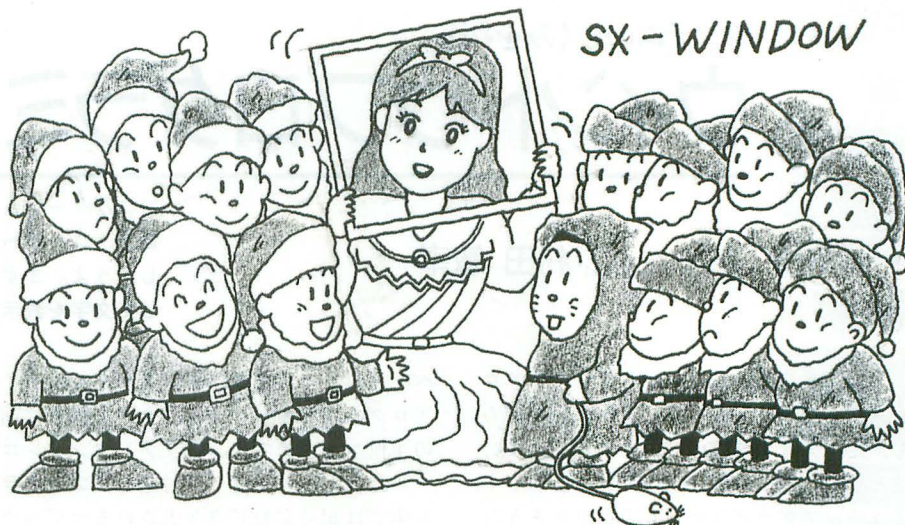
KEYBOARD：キーボードマン

キーボードマネージャ。キーボード関係の制御をするのだが，特に彼の世話になることはないだろう。縁の下グループの一員だ。

KEY：キーマン

キーマネージャ。キーマンというほどの者ではない。キーが押されるとキーボードマンからそのデータを受け取り，キーが押されたイベントを発生するのが仕事だ。彼も縁の下グループの一員。

EVENT：イベントマン

イベントマネージャ。別に地方イベント担当の電通社員ではない。イベントというのはマウスがクリックしたりキーが押されたりウィンドウが開いたり動いたりといった出来事全部のこと。実のところ，タスクマンの下働きをしている。

RESOURCE：リソースマン

リソースマネージャ。リソースというデータを管理する人。プログラムで扱うデータをリソースってことにして，独立させて，管理してしまおう。

GRAPHIC：グラフマン

グラフィックマネージャ。ブラフマンはインドはバラモン思想の中心概念。グラフマンはSX-WINDOWにおける描画の中心。SXシェルの外からでも使える。グラフィックといっても，テキスト画面にも描画できる。

WINDOW：ウィンドウマン

ウィンドウマネージャ。ウィンドウ作成や描画や消去を管理する。お世話になります。

MENU：メニューマン

メニューマネージャ。メニューというのは，お馴染みポップアップメニューのこと。必要なデータを彼に渡してやると，ポップアップメニューしてくれる。

CONTROL：コントロールマン

コントロールマネージャ。コントロールというのは，ウィンドウをコントロールするアイテムのこと。具体的には，ボタンとかスクロールバーとか，そういったものだ。

DIALOG：ダイアログマン

ダイアログマネージャ。ダイアログというのは対話ボックス。ウィンドウのように動かしたり閉じたりできず，とにかく指示に従った操作をしない限り画面に居座ったままのやつのこと。たとえば，ディスクコピーやフォーマット時のサーキットの絵はダイアログだ。

TEXT：テキストマン

テキストマネージャ。まあ，テキスト入出力をやってくれるものだ。

TASK：タスクマン

タスクマネージャ。メモリマンと並んで，SXシステムの基幹を担う大御所。マルチタスクの素。たいていのアプリケーションは彼の下で動く。

* * *

というわけだな。それぞれが機能分担して働き，プログラムを動かす。SX-WINDOW上でプログラムが正しく動くのは，マネージャさんが働くからといって過言ではない。ライブラリにはこいつらを働かせるためのルーチンが入っているわけだ。非常にたくさんマネージャさんがいる。そのため，新しく覚えなければならないことも多い。マネージャさんはその分，働き者なのである。

OSに管理されていないプログラムに慣れた人はどうしてこんな面倒なことをしなければならないのかと思うかもしれない。が，面倒だからと構造体やポインタを使わずC言語を使う人はいない。最初は慣れにくい概念でも，それが高次元な処理を実現してくれるからだ。SX-WINDOWのマネージャたちが提供するサービスはかなり高度なものがある。それは，慣れてしまえば，これまでのプログラミングがひどく野蛮なものに見えるかもしれないと思わせるほどのものである。

いまはまだ，膨大な資料に圧倒されていてもいい。さらに詳しくは以降の記事や，付録ディスクに記載されているのでそちらをどうぞ。これから何カ月かにわたってSX-WINDOWの特集が続く予定だ。少しずつものにしていってほしい。

システムのしくみを探る

ウィンドウプログラミングへの道

Murata Toshiyuki 村田 敏幸

アプリケーションのないウィンドウシステムなんてとお嘆きの貴方，ユーザーだったら待っていないで自分でもなんとかしましょうよ。まずはその第1歩。基礎概念から，ウィンドウを出して文字を表示するまでを一気に走ってみてください。

発表以来，高いところから「ウィンドウシステムだぞ，えっへん」とかいいながらも，ただ大きくて遅いビジュアルシェルに甘んじていたSX-WINDOWが，ようやく，アマチュアプログラマにも手を出せそうなところにまでおりてきました。お待ちかねの，技術資料公開です。今月号の付録ディスクに，ドキュメント（以下，嘘を承知で“簡易マニュアル”と呼んでしまいます），開発関連ツール，サンプルからなる開発キットが収録されていますから，まずは解凍して，その分量に圧倒されてください。若干，ショックが和らいで，興味がむくむく頭をもたげてきたところで，本文へどうぞ。アセンブリ言語レベルで「ウィンドウを開いて文字を表示するプログラム」を作ることを目標に，周辺から用語と概念の解説をたんたんとやります。

なお，これからする話のほとんどは簡易マニュアルを読めば書いてあることばかりです。いうまでもないことですが，マニュアルは無精せずにきちんと読みましょう（比較的読みやすく書かれていることでもありますし）。ところで，世のマニュアルには“読むマニュアル”と“引くマニュアル”の2種類があるのはご存じですね。付録ディスク中のドキュメントファイルでは，～.DOCが読むマニュアル，～.REF，～.LSTが引くマニュアルです。

参考書

最初に市販の参考書の紹介からまいりましょう。とはいっても，SX-WINDOWの解説書なんかまだどこにもありません。Macintoshの本を参考にしてしまおうという話なのです。噂されていたように，そして，解析してみた方ならすでにおわかりのように，SX-WINDOWはかなりの部分，MacintoshのToolboxをパク，いえ，意識して作られています。グラフマネージャはQuickDrawに生き写し，テキストマネージャはTextEditのそっくりさん，メモリ，イベント，ウィンドウ，コントロール，ダイアログの各マネージャも完全コンパチ寸前の下位コンパチです。各ファンクションコールはほとんど1対1に対応し，プログラム中では記号定数で書き表されるマジックナンバーまでわざわざ同じ値に揃えてあります。

あんまり似ているので，ついつい知的所有権のことが気になってしまいますが，もちろん，SX-WINDOWのコード自体はシャープのオリジナルで，もし，Appleに訴えられても勝てるだけの証拠が保管されているはずです。私たちはそんなことはまったく心配せずに，「シャープさんが私たちの便宜のためにMacintoshに似せてくれたのだ」と思って，利用できるものは利用しプログラミングに励みましょう。

●参考書を利用するうえでの予備知識

Macintoshの解説書を利用するときには，どこまでがSX-WINDOWと同じで，どこから違うのかを見極めながら読まなければなりません。MS-DOSや8086用のCの本を，Human68kやXCプログラミングの参考にするときと同じ脳味噌の使い方が必要です。簡単に要点を挙げておきます。

1) 概念は共通
2) 用語はほとんど同じ（異なるものについてはあとでちょこちょこと触れます）
3) ファンクションコール名，コール番号はぜんぜん違う（ちょっと痛い）

また，Macintoshの主力開発言語はPascalで（最近はCも広く使われているようですが），解説書もPascal主体で書かれていますから，Pascalの知識もあったほうがよいでしょう。もっとも，Cを知っていれば，

```
Integer     →short
Longint     →long
Nil         →NULL
Boolean     →enum { FALSE,
                    TRUE, }
Record      →struct
Function foo(~):Integer
            →short foo(~)
Procedure proc(~)
            →void proc(~)
```

などなどとフィーリングで読み替えればなんとかなります，きっと。

●Inside Macintosh

Macintoshの資料といったら，なんといっても，Toolboxの開発者らの手による標準マニュアル『Inside Macintosh(Addison-Wesley)』です。日本語版はトッパンからVolume I&II, III&IV, Vの3冊で発売されています。ぶ厚い百科事典のような本です。それぞれ，20,085円，12,875円，13,900円（税込み定価）ですから，合計するとC compiler PRO-68K Ver.2.0の定価を越えることになります。もっとも，SX-WINDOW上のプログラミングの参考にするだけであれば，I&IIだけがあればよいでしょう。III&IVはMacintoshのハードの話＋I～IIIのサマリー総集編＋いわゆるFat Mac以降の拡張部分という構成ですし，VはMacII以降の拡張部分の補足解説にすぎず，Color QuickDrawについて書かれた数ページが僅かに関係あるかな？　という程度です。本当のところはIIもメモリマネージャの部分以外は役には立たないようですが，IとIIで1冊ですからしかたがないですね（それとも原書に手を出しますか？）。

さすがに内容は充実しています。ほかの解説書では割愛されている深い部分に関しては，この本に頼るよりありません。マニュアル臭をぷんぷんさせ，サンプルプログラムはおろか，コール例もほとんどないのが珠に傷ですが，豊富な図版が理解を助けてくれるでしょう。また，各章末のサマリー部分には必ずPascalでの使い方と，アセンブリ言語レベルでの使い方の両方が載っており，リファレンスとしても申し分ありません（本の厚さのせいで引きにくいのは確かですが）。ただ，文中の固有名詞はともかく，サマリー中のコメントまで英語のままですので，「どこが日本語版なんだ！」という憤りを感じることはあるかもしれませんね。

●マッキントッシュの道具箱

もう少しお手軽な本では『マッキントッシュの道具箱(パーソナルメディア)』があります。Vol.I, IIの2冊で各4,944円(税込み定価)です。Vol.Iがメモリマネージャとquickdrawなどの基本部分，IIがウィンドウ関係，という構成になっています。著者は『Inside Macintosh』にも関わった人のようで，その点，信頼できます。各章末のリファレンスも感じよくまとまっていて，とくにワンポイントアドバイス的なノート部分は簡潔に要点を示してくれています。コール例+α程度のサンプルプログラムも多数用意されており，とくにVol.IIの巻末にはテキストエディタの全ソースも載っています。ただし，ソース中のコメントは訳されていません。あ，訳といえば，私はこの本の訳があまり気に入っていないということをつけ加えておきましょう。

『マッキントッシュの道具箱』には，『続マッキントッシュの道具箱』という続編もあります。もっとも，こちらは著者・訳者とも別人で，本の性格も異なります。Macintosh上でプログラムを作る際の注意点やデバッグ技法について述べられた読み物です。資料性の高い本ではありませんが，メモリ管理絡みの話などは参考になるでしょう。SX-WINDOWには関係ない部分も，「あっちのプログラマはこうやってデバッグしているんだな(同じだな，違うな，いいな，なにそれ)」と考えながら読めば，それなりに楽しめます。

●インサイドマック徹底ガイド

さらに手軽なところでは，『インサイドマック徹底ガイド(BNN)』があります。上下巻，各3,300円(税込み定価)です。価格も手ごろで，内容もよくまとまっており，なかなかのお勧め品といえましょう。上巻だけでも十分役に立ちます。というより，下巻はほとんどのページをMacintoshのOSに割いているため，あまりSX-WINDOWには関係ないのです。

「How To Use Inside Mac.」という副題が示すように，内容は『Inside Macintosh』の原書を読むためのガイドといった感じになっています(そういいきってしまうのは失礼かもしれませんが)。『Inside Macintosh』から必要最小限の部分を抜き出してきたような作りです。その分，エッセンスが詰まっているともいえます。部分的に抄訳っぽい部分も見られるものの，基本的には純国産ですから，訳の悪さに煩わされることもありません。小中規模のサンプルも適当にちりばめられており，上巻の巻末には大きなサンプル(テキストエディタ)の全ソースも載っています。

●雑誌記事

雑誌記事では，『インターフェース』誌1988年12月号の特集，“Macintoshの「技術」徹底解剖”がかなり参考になりました。サンプルプログラム(またしてもテキストエディタ)がCで書いてあるのが見どころです。

また，私は雑誌の色が嫌いなのでまじめに読んだことはありませんが，Mac専門各誌にも目を通してみれば，なにか得るものがあるでしょう。

SX-WINDOWシステムの概要

プログラミングに入る前に，周辺の話題を雑学風(?)に取り上げてみます。

●バージョン

SX-WINDOWにはEXPERT II/PRO IIに添付されたVer.1.00，SUPER-HDに添付されたVer.1.01，そして，パッケージ版のVer.1.02があります。これだけ頻繁にバージョンアップしたところをみると，ぼろぼろとバグが出たのでしょう(と，知らないふりをする)。エンバグしてなければいいですね(と，知らないふりをする)。Ver.1.01以前の登録ユーザーに対しては無償でバージョンアップサービスが行われたようですから，すでに読者の手元にはVer.1.02があることと思いますが，もし，まだ古いバージョンを使っているようであれば，早急に手を打つことをお勧めします。バージョンアップの過程では，たんなるバグとりだけではなく，処理速度の向上や，機能の追加もあわせて行われているからです。

●システムとしての位置づけ

SX-WINDOWシステムの本体はFSX.Xという1本のプログラムで供給されています。FSX.XはCONFIG.SYSにより組み込まれるデバイスドライバとして，また，起動後にコマンドラインから実行する常駐プログラムとしてメモリに居座り，Human68kに張りついて，ウィンドウシステムとしての機能を付加します。見掛けは，ほかのデバイスドライバの類，たとえばFLOATn.XがHuman68kに浮動小数点演算機能をつけくわえるのに似ています。しかし，FLOATn.XがコブのようにHuman68kにくっつくのに対し，SX-WINDOWはHuman68kをすっぽり包み，さらにはIOCSやハードの一部にまで触手を伸ばして管理下に置き，全体をひとつのシステムにまとめ上げるのです。

シェルであるSXWIN.X(以下，簡易マニュアルに従いSXシェルと呼びます)も，SX-WINDOWシステム上の1アプリケーションにすぎません。ちょうど，VS.XがHuman68kのアプリケーションのひとつだったようにです。その気になれば，べつのシェルを作って差し替えることも可能ですし，SXシェルを使わずに単独で動くプログラムを作ることもできます。

ところで，コマンドラインから組み込んだFSX.Xは，もう一度FSX.Xを起動することで常駐解除することができます。常駐プログラムは/rスイッチによって切り離すように作るのがふつうですが，FSX.Xはスイッチなしなのですね。私は意地が悪いので，それが何を意味するのか知りつつ，SXシェル上からFSX.Xをダブルクリックして喜ぶのでした(こういう小さな部分でシステムの信頼性を計るのは間違いだと思いますか?)。

●SXコール

SX-WINDOWのファンクションコール(以下，簡易マニュアルに従いSXコールと呼びます)はA系列未実装命令を使って呼び出します。Human68kのDOSコールやFLOATn.XのファンクションコールはF系列の未実装命令を使って呼び出すわけですが，$Fxxx_H$の代わりに$Axxx_H$を使うこと以外はSXコールも似たようなものです。引数もDOSコール同様，スタックに積んで渡します。

SXコールは公開されているものだけでも500個弱あります。Human68kのDOSコールが100個ちょっと，IOCSコールが200個程度ということを考えると，とんでもない数です。しかも，公開されていないものがこのほかにもまだあり，ざっと数えてみただけでも100個以上のSXコールが未公開のまま眠っています。なかには，バグのため使えないものとか，“コーヒーのないクリープ(誤植じゃないぞ)”のごとき無意味なものもあるようですが，せめて既存のデスクアクセサリなどで使用しているSXコールはすべて公開してほしかったところです。

●イベントドリブン

イベントドリブン(event driven:イベント駆動)とは，プログラムがなんらかの出来事(=イベントの発生)をきっかけとして駆動される様子を表す言葉です。イベントが発生するのをじっと待ち，発生したらそのイベントの種類に応じた処理を行い，また次のイベントを待つ，そのように作られたプログラムのことをイベントドリブン

型のプログラムと呼んだりもします。

ここでいうイベントとは，プログラムに対する外部からの働きかけの総称です。主にキー入力やマウスボタンのON/OFFといったユーザーのアクションを指しますが，もっと抽象的なもの，内部的なものもイベントの範疇に含みます。抽象的なイベントとしては，“イベントがなにもないことを表す”イベントとか，システムからの合図の類などが考えられます。プログラムにとってはユーザーのアクションもシステムからの合図も外部からの働きかけには違いないわけです。

イベントドリブンの考え方は，複数のイベントが非同期的に起こるような状況でとくに有用になってきます。

第1に，プログラムの構造がすっきりします。イベントドリブン型プログラムのメインルーチンは，

1) イベントを取得する

2) イベントの種類に応じて処理を振り分ける

3) 1)に戻る

という，非常にシンプルなループの形にまとめることができるのです。しかも，イベントドリブン型プログラムを前提にしたシステムでは，1)のイベントを取得する部分がシステムコールとして提供されるのがふつうですから（もちろんSX-WINDOWもそうです），プログラム側の負担はさらに軽くなります。個別にキースキャンしたり，マウスのボタンの状態をチェックしたりといったことをしなくても，システムに漠然とイベントを要求すれば，“このキーが押された”とか“マウスの左ボタンが押された”といったイベントが返ってくるわけです。

プログラムをイベントドリブン型にする第2のメリットは，ユーザーにとってのものです。イベントドリブン型プログラムでは，基本的にすべてのイベントが対等に扱われます。イベントに対応する処理ルーチンはおのおの独立していますから，呼び出す順序＝イベントの発生順序＝ユーザーのオペレーションの順序にも制限がありません。

ウィンドウシステム上では，マウスボタンが押されたり離されたり，キーが押されたりといったイベントがユーザーの気の向くままランダムに発生します。プログラム側はそれに備えなければなりませんし，ユーザー側もそれを期待します。となると，ウィンドウシステムにイベントドリブンという組み合わせは，プログラムを作る側にも使う側にもなかなかおいしい選択といえるのではないでしょうか。

図1　マネージャの階層（だいたいこんな感じ）

<table>
<tr><td colspan="3">タスク</td></tr>
<tr><td colspan="3">ダイアログ</td></tr>
<tr><td colspan="3">ウィンドウ</td></tr>
<tr><td colspan="3">コントロール</td></tr>
<tr><td colspan="3">テキスト</td></tr>
<tr><td colspan="3">メニュー</td></tr>
<tr><td>グラフ</td><td colspan="2">イベント</td></tr>
<tr><td>リソース</td><td>キー</td><td>マウス</td></tr>
<tr><td rowspan="2">メモリ</td><td rowspan="2">キーボード</td><td>アニメーション</td></tr>
<tr><td>エクセプション</td></tr>
</table>

●イベントドリブン型の疑似マルチタスク

イベントドリブンは直接マルチタスクに結びつく概念ではありません。現にMacintoshのシステム（ただし，Multi Finderなしの標準状態）はシングルタスクでありながら，各プログラムはイベントドリブン型をしています。両者をつなぎあわせるのには多少の小細工が必要になってきます。

ところで，マルチタスクという言葉は説明する必要ありませんね？　複数のタスク（task：仕事・処理の単位）が並行動作している（ようにみえる）状態を指す言葉です。プロセッサがひとつしかないふつうのコンピュータでは，タスクを少し動かしては止め，また別のタスクを動かして，というようにシステムが頑張ることでマルチタスクが実現されます。ちなみに，タスクに類した単語に，ジョブ（job），プロセス（process），多少ニュアンスは違いますがスレッド（thread）などがあります。英語ではそれぞれ“仕事”とか“一連の処理”といった程度のぶっきらぼうな言葉です。根本的な意味の違いはないと考えてよいでしょう。ただ，システムによって，仕事・処理の単位にいろいろと呼び名をつけて区別しているのです。余談でした。

では，イベントドリブン型のプログラムがマルチタスクで走るためにはなにが必要で，実現するにはどうしたらよいか，考えてみましょう。

第1に，タスクをいつどこで切り替えるかという問題があります。これは比較的簡単に解決できそうです。各タスクはメインループの中で1回，システムにイベントを要求してきます。タスク側から見れば，単にシステムコールを発行したというだけのことですが，システム側から見れば，自分に制御が返ってきたことになります。ここにタスクを切り替える隙があります。タスクAから戻ってきたときに，タスクBにイベントを返し，タスクBから戻ってきたら，タスクAにイベントを返す，という具合です。

第2に，十分短い時間ごとに，各タスクに制御を渡してやらなければなりません。これは，イベントが発生するまでじっと待つイベントドリブン型プログラムとは相性が悪いようにみえます。しかし，逆にいうと，イベントが頻繁に発生しさえすれば，タスクに制御が移る回数も多くなるわけですから，意味のない架空のイベントを作り上げるという案が浮かびます。ヌルイベント，つまり“イベントがないことを表すイベント”です。イベントが発生していないときには，システムは適当な間隔で，このイベントを返し，眠っていたタスクを起こします。タスクはしばらく走って，また，次のイベントを要求するわけです。

そして，この2つの案を組み合わせると，SX-WINDOWのイベントドリブン型の疑似マルチタスクになります。“疑似”がついているのは，タスク側の協力がないとマルチタスクにならない，という意味にとりましょう。制御を渡したが最後，システムにイベントを要求しないプログラムがひとつあるだけで，マルチタスキングはぴったり止まってしまいます。そこまで酷くなくても，のろまなタスクがいれば，システム全体の速度ががたんと落ちます。システム側には，制御を強制的に自分に引き戻す手段がないのです。

SX-WINDOWの各マネージャ

マネージャとは，なにか特定の分野を一括して管理するプログラム/ルーチン集をいいます。SX-WINDOWシステムも，いくつかのマネージャから構成されています。もっとも，プログラム自体はFSX.Xひとつにまとめられているのはご承知のとおりです。マネージャという呼び名は論理的な区分けにすぎません。

SX-WINDOWシステムには，15のマネージャがあります。マネージャには低水準なものから高水準なものまで，数段階のランクがつけられています（部分的には曖昧なところもありますが）。いちおう図1に各マネージャの階層を示しておきます。図中，上にいくほど高レベル・抽象的になり，同時に下位のマネージャに依存する度合いが増す，と思ってください。

では，以下，各マネージャの働きを簡単に紹介しましょう。重要なものについては，あとでまた振り返ります。

●メモリマネージャ

メモリの総括的な管理を行います。上位のマネージャやアプリケーションプログラムは，メモリが必要になったらメモリマネージャにメモリを要求し，確保してから，使うことになります。

●エクセプションマネージャ

非常に低水準な，マネージャと呼ぶのもおこがましいようなマネージャです。ひとつのエクセプション（exception：例外）に対して，複数の処理ルーチンをぶら下げることができるよう細工します。もっとも，いまのところは，垂直帰線信号（V-DISP）による割り込み専用です。SX-WINDOWはシステム時間を計るタイマに垂直帰線信号を使っています。

●マウスマネージャ

マウスの移動，マウスカーソルの表示を担当します。マウスデータを受信するところまではIOCS同様にROMの割り込みルーチンを使用し，受信したデータだけを盗んできて細工を加えているようです。これにより，マウスの移動量や，移動方向の左右をソフト的に変更することができるようになっています。

●アニメーションマネージャ

垂直帰線信号による割り込みを用いてマウスカーソルのアニメーションを行うだけのマネージャです。簡易マニュアルによればエクセプションマネージャの一部だそうです（さもありなん）。

●キーボードマネージャ

キーボードからキーデータを得て，キーバッファに溜め込みます。やっていることはROMのキー入力時割り込みルーチンとさほど変わりません。また，提供されるサービスもIOCSのキー入力関係コール程度のものです。が，XF1～XF5をシフトキーとみなすといった細工が施されているのが面白いところです。

●キーマネージャ

キーボードマネージャから生のキーデータを受け取り，CONデバイスを通してASCIIコードに変換します。これは日本語入力FEPを有効にするための処置です。

●イベントマネージャ

イベントの管理をします。イベントマネージャは，一部ウィンドウ関係のイベントも扱いますが，マウス，キー関係マネージャの元締めだと思っていればよいでしょう。マウスボタンの押し下げやキー入力をイベントとして内部のキュー（待ち行列）にため込みます。ため込んだイベントはのちにアプリケーションや上位マネージャから要求があったときに，ひとつずつキューから取り出されることになります。SXシェル上でマウスボタンを続けざまに何度も押すと，押した回数だけメニューがちかちか点滅するといった症状が見られますが，これはマウスボタンのON/OFFがイベントとしてキューにたまった結果です。

なお，イベントマネージャにイベントを要求したときにはタスクの切り替えは行われません。タスクの切り替えが起きるのは，タスクマネージャにイベントを要求したときです。

タスクマネージャはイベントマネージャからイベントを取得し，タスクマネージャ独自のイベントとともにアプリケーションプログラムに渡します。逆にいうと，タスクが切り替わってほしくない一時的なマウスボタンのチェック/キー入力のときにはタスクマネージャを通さずに，イベントマネージャに要求すればよいのです。

●リソースマネージャ

リソース（resource）の管理をします。リソースというのは……，あとでねっとり説明しますね。

●グラフマネージャ

SX-WINDOWシステム上のほとんどの画面描画を担当する，高級なテキスト/グラフィック描画ルーチン集です。MacintoshのQuickDrawに相当します。多くの場合，テキスト画面とグラフィック画面のどちらに対しても同様の描画が行えます。現行のSXシェルでは画面モードの関係でグラフィック画面は16色固定ですが，グラフマネージャ自体は，きちんと256色，65536色モードもサポートしています。65536色モードではColor QuickDraw似の色の混ぜ合わせもできるようです。

なお，Macintoshでいうフォントマネージャは，機能を大幅に縮小した形でグラフマネージャに併合されています。

●メニューマネージャ

ポップアップメニューの管理をします。Macintoshに比べるとかなり小規模なマネージャになっており，マネージャ自身を初期化する，リソースファイルからメニューを読み込んで使える状態にする，マウスの動きを追跡してメニューを選択する，の3つのファンクションコールしか用意されていません。実際の使用にあたっては，多少周辺ライブラリを整備してやる必要がありそうに思います。

●テキストマネージャ

テキストマネージャは，MacintoshのTextEditに相当し，基本的なテキスト編集機能を提供します。先ほど紹介したMacintoshの本でサンプルがみなテキストエディタだったのは，TextEditの存在ゆえです。SX-WINDOWのノート.Xも多分テキストマネージャを使って作られていると思われます（確認はしていません）。また，テキストマネージャはファイル名の入力などに広く利用されています（簡易マニュアルによると，そちらが本職だそうです）。

●コントロールマネージャ

コントロールというのは，ウィンドウ上に配置されたスクロールバーや各種ボタンの総称です。コントロールマネージャはそれらの制御をします。

●ウィンドウマネージャ

ウィンドウを管理・制御します。

●ダイアログマネージャ

ダイアログ（dialog box）は“対話ボックス”とか，“対話型ウィンドウ”などと訳されます。ユーザーに指示や確認を求めたりといった目的に使われる対話用のウィンドウのことです。ダイアログマネージャはこのダイアログの管理・制御をします。ダイアログ上には多くの場合，さまざまなコントロール類が置かれますが，ダイアログマネージャはこのコントロールの面倒もみてくれます。

なお，ダイアログはウィンドウの特別な形ということもあり，ダイアログに対してもウィンドウマネージャのファンクションを適用できます（そのはずです）。

●タスクマネージャ

タスクの管理をするSX-WINDOWシステムの総元締めです。Macintoshでいうスクラップマネージャやファイルマネージャの機能も包含しています。

メモリ関連

そろそろプログラミングの準備に入りましょう。まずはメモリマネージャによるメモリ管理関連を押さえておきます。

メモリマネージャはメモリをヒープ（heap）の形で管理します。ヒープは1個以上のメモリブロックからなり，メモリ確保要求があるごとに，メモリマネージャは要求バイト数を満たす空きメモリブロックを検索し，その一部ないしはすべてを確保します。なお，メモリマネージャは複数のヒープを管理することもでき，現在使用中のヒープをカレントヒープと呼びます（複

数のヒープを使う場面はあまりないと思われますが)。

メモリブロックにはいくつかの属性・種類があります。まず，少なくとも，各ブロックは割り当て済み (allocated block) か未割り当て (free block) かのどちらかです。さらに，割り当て済みのブロックは図2のようないくつかの種類に分類されます。

リロケータブル (再配置可能：relocatable) なブロックは，メモリマネージャによってヒープ内を移動させられる可能性のあるブロックです。メモリの確保・解放を繰り返すことでヒープは図3のa)に示すように細切れに細分化される場合があります。一度に確保できるメモリブロックの最大の大きさは，最大の空きメモリブロックの大きさに制限されますから，空きメモリの総和が十分あっても必要なだけのメモリブロックが確保できないという事態が生じることになります。そこで，メモリマネージャは図3のb)のようにリロケータブルなメモリブロックを動かし，詰めて，要求されただけの大きさのメモリブロックを作ろうとするわけです。このメモリブロックを詰める操作をコンパクション (compaction：コンパクト化) といいます。

さて，コンパクションの発生によって，リロケータブルなブロックは勝手にどこかに動かされます。その結果，図4のように，このブロックを指していたポインタは意味をなさなくなってしまいます。これではまともなアクセスはできませんね。というわけで，実際にはリロケータブルブロックはポインタではなく，ハンドル (handle) によって参照されます。

ハンドルというのは“何かを間接的に表現する代名詞”を意味する用語ですが，ここでいうハンドルは“リロケータブルブロックを指すポインタ (＝マスタポインタ：master pointer)を指すポインタ”，のことです。マスタポインタはヒープ上の固定領域にいくつもまとめて置かれ，個々のマスタポインタはリロケータブルブロックを指しています。そして，コンパクションが発生したときには，マスタポインタも同時に

図2 メモリブロックの種類

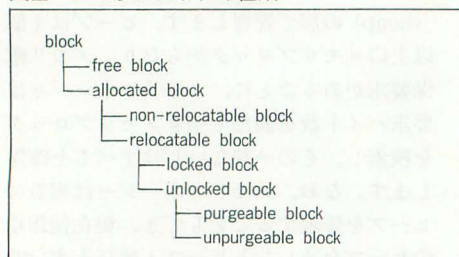



図3 コンパクション

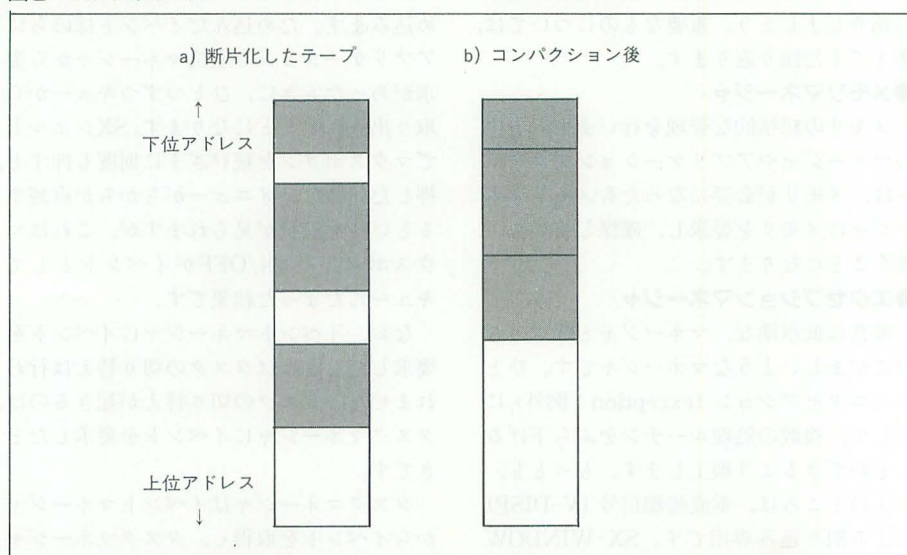



図4 ハンドルとマスタポインタ

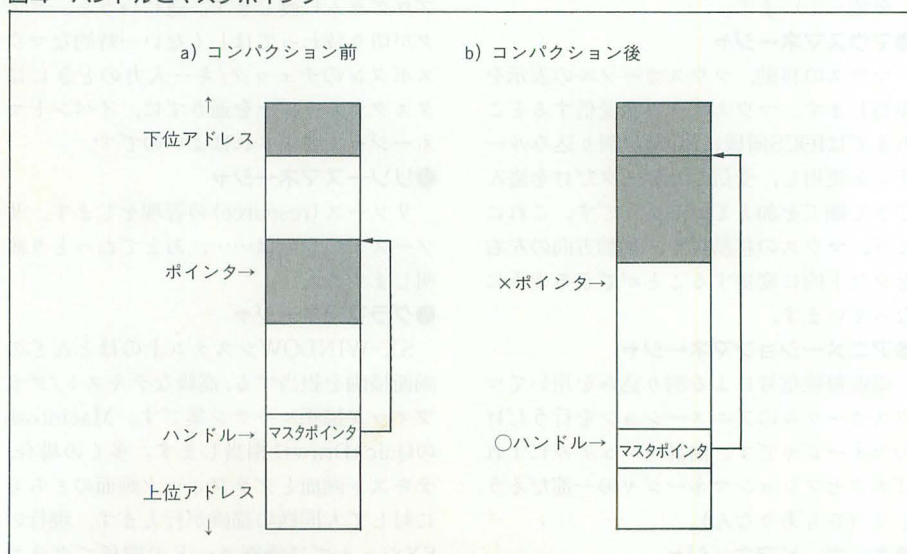
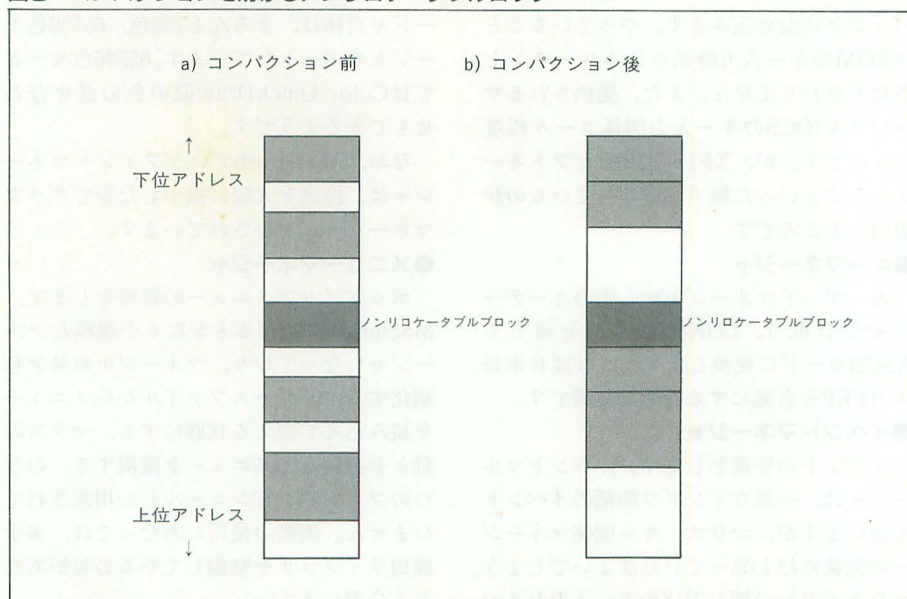



図5 コンパクションを妨げるノンリロケータブルロック

書き換えられ，つねに正しくリロケータブルブロックを指すよう保たれます。
1) マスタポインタは動かない
2) マスタポインタはつねにリロケータブルブロックを指している
の2点から，ハンドル（マスタポインタへのポインタですよ）に2重間接を適用することにより，いつでもリロケータブルブロックへのアクセスが可能となるわけです。図4にそのイメージを示します。

対して，ノンリロケータブル（non-relocatable：再配置不可）なメモリブロックは，ヒープ上に固定され，動くことのないメモリブロックです。ノンリロケータブルなブロックは，ポインタによって参照することが可能であり，ハンドルを介するリロケータブルブロックよりはアクセス速度・手間の点で有利ではあります。が，図5に示すようにノンリロケータブルブロックはコンパクションの妨げとなりますので（メモリマネージャはほかのブロックをまたいでブロックを動かすことはありません），効率的にメモリを使おうと思ったら，なるべくノンリロケータブルブロックは使用しないようにすべきです。

さて，リロケータブルブロックは，もし望むのであれば，コンパクションの対象から外すこと，つまり，その位置を固定することができます。位置を固定してしまえば，マスタポインタのコピーを作ってアクセス速度を稼ぐことが可能になるわけです。ただし，固定された（locked）メモリブロックは，やはりコンパクションの妨げとなりますから，アクセスが済んだら忘れずに移動可能（unlocked）に戻す必要があります。なお，リロケータブルブロックの初期状態はunlockedです。

またさて。ときには，コンパクト化してもなお，メモリが足りない場合があります。それでも，メモリマネージャはあきらめません。リロケータブルで，かつ，固定もされていないブロックの中から，不用なものを削除し始めます。これをパージ（purge）といいます。もちろん，手当たり次第にメモリブロックを捨てられてはたまったものではありませんね。メモリマネージャがパージの対象にするのは，パージ可能（purgeable）という属性がついたメモリブロックだけです。リロケータブルブロックの初期状態はパージ不可（unpurgeable）になっていますから，アプリケーション側で故意にパージ可能にしないかぎり，勝手に消されたりはしません。

プログラムの実行に必要なデータなどをディスクからメモリに読み込んで使う場合を考えましょう。そのデータは，参照される瞬間だけメモリ上に存在すればよいわけですから，極端な話，使いたいときに毎回ロードしてもよいことになります。が，それではロード時間が馬鹿になりません。といって，使わないときにもメモリ上にデータが居座っていると，それだけメモリを圧迫します。こういうときには，そのデータを置くメモリブロックをパージ可能にしておくのが好ましいといえます。データを参照したいときに，パージされてしまっているかどうか調べ，まだ生きていればそのまま使い，さもなくばメモリを再確保してディスクからデータを読み込み直すというわけです。

なお，リロケータブルブロックがパージされたときには，マスタポインタは0に書き換えられます。マスタポインタが0かどうかで，パージされたかどうかがわかるのです。値が0のマスタポインタを指す空のハンドル（empty handle）は，専用のファンクションコールを使って再割り当てして使います。

リソース

●SX-WINDOWにおけるリソース

簡易マニュアル中，RESOURCE.DOCは，“SX-WINDOWでいうリソースはMacintoshのそれとはまったく別物である”，という断り書きから始まっています。確かに，ファイルシステム自体が大きく異なるということもあり，Macintoshとはリソースの実現・管理方法が違います。ですが，部分的には非常に忠実にMacintoshのリソースをパク，いえ，模倣しているところもあり，参照のしかたなども似たようなものです。管理方法だけを指して“まったく別物”というのはちょっと言いすぎです。では，どこが違うのでしょう。

つらつら考えますに，たぶん，根本的なスタンスが違うのです。ポリシーの有無ともいえます。Macintoshのリソースはある意味でシステムの中核を成すのに対して，SX-WINDOWのそれは“主”ではなく“従”なのですね。積極的に，アプリケーションにリソースを利用してもらおうという姿勢が最初からありません。そう考えていくと，アイコンがエディットできないといった点にも合点がいきます。

●リソースとは

さて，そもそもリソースとはなんなのか，まだ話していませんでした。簡易マニュアルでは，“ファイル名の代わりにリソースタイプとリソースIDによって参照される特殊な形式のファイル”といっています。SX-WINDOWのリソースは，ちょうどアーカイブファイルかライブラリファイルのように，拡張子が.LBのファイルにいくつも詰めて格納されますから，そのような言い方もできるでしょう。でも，なんかピンときませんね。

多少嘘混じりで翻訳しますと，プログラムの実行に必要なデータや補助的な実行コードを，メインプログラム（のファイル）から分離したもの，それがリソースです。いわゆる，データファイルや，オーバーレイされるサブモジュール，ダイナミックリンクされるライブラリ的なモジュールなどは，みんなリソースとなりえます。

そうするとどのようなメリットがあるのでしょう。まず，プログラムを作る側からいえば，データなどの細かな修正や追加時にもプログラム本体を再アセンブル・再コンパイルする必要がなくなります。これは，場合によっては，ユーザーレベルでプログラムの部分的なカスタマイズができることを意味します。また，分離されたリソースは必要なら複数のプログラムで共用することもできます。まさに，資源なのです。

もっとも，プログラム本体とリソースを分離することによって管理が繁雑になるのもまた事実です。SX-WINDOWでは複数のリソースをひとつのリソースファイルにまとめておくことによって，辛うじてこの問題を回避しています（Xファイルを拡張するといった案は出なかったのですか?）。ちなみに，Macintoshの場合は，ファイルシステムそのものがリソースを扱う前提で設計されています。簡単にいうと，“プログラムとリソースは物理的には分離されているが，論理的にはひとまとめ”です。

●リソースタイプとリソースID

リソースは，リソースタイプとリソースIDの2つの数値によって区別・識別されます。リソースタイプは4バイトからなり，通常，意味を持った（そして読める）4文字のASCIIコード列が使われます。たとえば，WDEFという名前で知られるリソースタイプの値は，W，D，E，Fそれぞれを ASCIIコードにして並べた，

57444546_H

となるわけです。

リソースIDのほうは，同種のリソースタイプ内での識別番号で，−32768〜+32767の2バイト符号付き数で表されます。識別番号ですから，同種のリソースタイプ内で

は同じIDを複数のリソースに割り当てることはできません。また，負の値と0～127まではシステム予約で，ユーザーアプリケーションで使用するリソースには128～32767の範囲のみが使えることになっています。

ところで，Macintoshでは，新しいリソースタイプ名が必要なときにはApple社にお伺いを立てて調整すると聞きます。SX-WINDOWではどうなるのでしょう。

●リソースの作成

リソースは，リソースエディタないしはリソースコンパイラで作成するものです。前者は，対話型のリソース作成・修正用のツール，後者は非対話型であらかじめリソースを簡易言語で書き表しておき，コンパイルすることでリソースを作り出すツールです。ですが，いまのところSX-WINDOWではそのどちらもサポートされておらず，リソースはアセンブラを使って作るしかありません。つまり，.dc疑似命令でリソースの内部構造を再現するわけですね（個々のリソースの具体的な構造はここでは触れません。SXCALL.MACないしはSXDEF.Hを参照してください)。状況はかなり悲惨です。

そのためかどうか，リソースを積極的に利用しているのはシステム本体とSXシェルだけで，デスクアクセサリあたりになると，ほとんどリソースを活用していません。唯一，コントロールパネルは，専用のリソースファイルCTRLPNL.LBを持ち，分離できるものはとことん分離した，なかなか模範的な作りになっています。

リソースエディタはともかく，リソースコンパイラぐらいなら，パワーユーザーの手の届く範囲だと思われます。公開ソフト市場に出回るのを待ちましょう（私はそれほど他力本願ではありませんけど)。

●リソースファイル管理ツール

リソースそのものを作るツールは未サポートながら，リソースファイルの管理ツールは今回ディスクに収録された開発キットの中に入っています。RLK.Xがそれです。どこにも解説がないようですが，コマンドラインから単に，

A>RLK

として起動すればヘルプが出ますから，すぐにでも使えるでしょう。アーカイバやライブラリアンと同じような使い勝手です。ただし，RLK.Xがサポートしているのは，

・リソースの更新・追加
・リソースの削除
・リソースの一覧表示

のみで，なぜか抽出ができません（解析魔封じ！)。公開ソフトに期待しましょう。

使用上の注意を少々。当然のことですが，システムが使用しているリソースをむやみに削ったり改変したりしてはいけません。自分がなにをやろうとしているのか，その結果なにが起きるか理解したうえで利用してください。少なくとも，リソースファイルのバックアップはとっておくべきです。もっとも，現状ではビープ音を差し替えて遊ぶぐらいの使い道しかないかもしれませんね。例のぴんぽーん音はリソースタイプBEEP，IDは0として，SYSTEM.LB中にあります。

もう1点。RLK.Xは非対話型のDOSコマンドですが，実行時にはFSX.Xが組み込まれている必要があります。

●SX-WINDOWのリソースファイルの中身

SX-WINDOWには4本のリソースファイルがついてきます。ここで，簡単に中身を紹介しておきましょう。上述のRLK.Xを使えば一覧表示を見ることができますから，各自，のぞいてみてくださいね。

SYSTEM.LBには，システム本体(一部SXシェル)で使うリソースが収められています。ウィンドウ，コントロール，メニューの種類に応じた制御・描画を行う実行コード(定義関数といいます)，ビープ音用のAD PCMデータ，プリンタマネージャらしき実行コードと複数のプリンタに対応するためのドライバ，SXシェルの背景描画用のグラフスクリプト（あとで説明します）などです。

BUILTIN.LBはSXシェル上のビルトインコマンド関係のリソースファイルです。ディレクトリを表示する例のウィンドウや，メニューから起動されるそのほかのプログラムの実行コード，また，各メニューのデータなどが収められています。

ICON.LBはアイコン関係です。アイコンのパターンや，アイコンとファイルとの対応を示したリソースなどが入っています。

CTRLPNL.LBは，先ほども出てきたようにコントロール.X用のリソースファイルです。アイコンのパターン，ダイアログテンプレート（ダイアログを開くときの引数をリソースにしたもの)，アイテムリスト（ダイアログ上のコントロールの種類・配置を示したリソースで，ダイアログテンプレートに結びついている)，さらには，各サブパネルそれぞれ用の実行コードが収められています。

ついでといってはなんですが，表1にSX-WINDOWで使っているリソースタイプの一覧を示しておきます（これで全部かどうかは自信がありませんが)。

ところで，SX-WINDOW Ver.1.01までは，リソースファイル中に開発過程のものと思われる余計なリソースがたくさん残っていました。とくにSYSTEM.LBはゴミの宝庫で，ウィンドウ定義関数はダブっているわ，正常動作しないウィンドウ定義関数やコントロール定義関数があるわで，私たちのディスクスペースを無駄遣いしてくれていたのです。これらをばっさり削った結果，Ver.1.02のSYSTEM.LBはVer.1.01よりも30Kバイトほどファイルサイズが

表1　リソースタイプ

タイプ	内容
ALRT	アラートテンプレート（使えないかもしれない）
BEEP	ビープ音AD PCMデータ
BGPT	背景パターンリスト（16×16ドット，4プレーン，8種類）
CDEF	コントロール定義関数
CMDS	ビルトインコマンドとファイル/リソースとの対応リスト
CNTL	コントロールテンプレート
CODE	各種実行コード
DITL	ダイアログアイテムリスト
DLOG	ダイアログテンプレート
DiME	ディレクトリウィンドウ用メニュー
ICN#	ファイルとアイコンの対応リスト
IMGS	イメージデータ
MDEF	メニュー定義関数
MENU	メニュー
PAT 3	モノクロ4階調パターンデータ
PAT 4	モノクロ4階調＋カラー3色パターンデータ
PICT	グラフスクリプト
PRTD	プリンタ機種別ドライバ
PRTM	プリンタマネージャ（？）
ShEV	SXシェル用データ（？）
ShME	SXシェル用メニュー
WDEF	ウィンドウ定義関数
WIND	ウィンドウテンプレート

小さくなっています。まずは，めでたし，めでたし。

●作成したリソースをどう使うか

簡易マニュアルには，既存のリソースファイルを利用する方法は記されています。リソースファイルをオープンしてから，リソースタイプとIDにより参照すればよいわけです。ところが，そこに至る道筋にはいっさい触れられていません（このあたりもユーザーにリソースを使わせようとしない姿勢の現れ，と判断します）。すでに述べたように，リソースはリソースコンパイラやリソースエディタ，または，アセンブラを使って作ります。できたリソースはRLK.Xを使ってリソースファイルにまとめます。どのリソースファイルに？　それが問題です。開発側で推奨する（あるいは想定している）方法が示されていないのです。

3案ほど考えました。ひとつは，既存のリソースファイルに収めてしまう方法です。SYSTEM.LBにでもいれておけば，SXシェル起動時に自動的にオープンされ，使える状態になります。ですが，システムのリソースと一緒にしてしまうのは，なんとなく気が引けます。

2案目は，プログラムそれぞれにリソースファイルを用意する方法です。コントロール.Xのように多くのリソースを使い，ほかのプログラムと共用するつもりもなければよい選択のように思えます。しかし，それほど多くのリソースを使わないのであれば，小さなリソースファイルがたくさんできる結果となり，管理が大変になります。

で，3案目です。自作プログラム用のリソースファイルを1本（たとえば，USER.LBといった名前で）用意して，その中にいろいろなプログラムのリソースを詰め込むという案です。結局，これが一番自然な方法なのでしょう。自作のプログラムでは起動時にこのリソースファイルをオープンするという，自分なりの約束を作るわけです。この方法にも問題がないわけではありませんが，話が具体的になりすぎそうなので，突っ込むのはまたの機会にします。

グラフマネージャ

●基礎概念

グラフマネージャは，“グラフ”と呼ばれる描画環境に対して描画を行います。ほんとうは少し違うのですが，グラフ＝ウィンドウと考えてみてください。グラフマネージャは個々のウィンドウのどれかひとつを描画対象にするわけです（現在描画対象になっているグラフをカレントグラフと呼びます）。グラフは構造体様のデータブロックで，その中身は，ビットマップ（へのポインタ），描画色や，描画モードなどといった，描画環境に関する情報から成ります。

ここで，ビットマップはグラフに結びつけられた仮想画面の種類や大きさ，アドレスを保持する構造体です。グラフマネージャにとっては，このビットマップで示されるメモリにデータを書き込むこと＝描画となります。“仮想”画面といってしまいましたが，通常，ビットマップはVRAMそのものを指しており，その意味では実画面です。ただ，グラフマネージャにとっては，VRAMであろうが，メインメモリ上に取られた仮想画面であろうが，たいした違いはありません。

グラフマネージャはテキスト画面とグラフィック画面をほとんど区別することなく扱えるようにできています。X68000のテキストVRAMはいわゆる水平型，グラフィックVRAMは垂直型と，両者の構造は異なりますが，グラフマネージャはビットマップにテキスト画面用のものとグラフィック画面用のものの2種類を用意することで，その違いを吸収します。ビットマップを差し替えるだけで，テキスト画面に対してもグラフィック画面に対しても，同様の描画が行えるのです。

●座標系

グラフマネージャは2種類の座標系を使い分けます。ひとつはグローバル座標であり，これは実画面上の絶対的な位置を表すものです。もうひとつはローカル座標で，これは各グラフごとの座標系です。またまた，嘘を承知でグラフ＝ウィンドウと考えてもらうと，ローカル座標は，ウィンドウの（タイトルバーを除いた）左上隅が(0, 0)になるような座標系です。グラフに対する描画時の座標指定にはローカル座標を使います。ローカル座標を使うかぎり，ウィンドウが実画面上のどこに表示されていようと，気にする必要はないわけです。

座標は上位ワードをx座標，下位ワードをy座標とする32ビットのデータで表現されます。ただし，グラフマネージャにおいてこの座標が意味するのはドットではなく，ポイントです。ポイントは大きさを持たない真の意味での点であり，各ドットの左上隅を指しています。対して，ドットは縦横1単位の面積を持ちます。

どうしてこうなっているかというと，MacintoshのQuickDrawがそうだったからです。では，QuickDrawがなぜこんな仕様になっているのかというと，画面上での図形の長さを数学的に表したかったからのようです。水平な線分の長さは，数学では両端点のx座標の差で求められます。ところが，X-BASICのline関数を考えてみると線分の長さは両端点のx座標の差よりも1ドット分長くなりますね。QuickDrawの開発者はこれを嫌い，座標の差がそのままドット数に反映されるようにしたのでした。

●描けるもの

グラフマネージャは，以下のようなさまざまな図形を描くことができます。

- ・直線
- ・長方形（rectangle：レクタングル）
- ・面取り長方形（rounded rectangle：角の丸まった長方形）
- ・真円を含む楕円（oval）
- ・自由な形の領域（regeon：リージョン）
- ・パターン（イメージ）
- ・文字，文字列

レクタングルなどの場合は輪郭（frame：フレーム）のみを描くか，塗り潰す（fill：フィル）かを選べるのはもちろん，いわゆるペンパターン，フィルパターン，描画モードを指定することもできます。

なお，MacintoshのQuickDrawでは，このほかに円弧と多角形，さらにシードフィルによるペイントがサポートされていますが，SX-WINDOWにはありません。一応“予約”になっていますから，そのうちサポートされるのでしょうか。

●レクタングル

繰り返します。レクタングルとは長方形のことです。「それなら長方形と書け！」と怒られそうですが，“××レクタングル”のような複合語を“××長方形”と表記するのでは間が抜けているので，以下，レクタングルで押し通します。

レクタングルは，左上隅の座標と右下隅の座標を表す2つのポイントを並べた形で表現されます。レクタングルの縦横の長さは座標の差でそのまま表されるのは前述のとおりです。左上と右下のx，y座標のどちらか少なくとも一方が等しい場合には，レクタングルは空になります（これをヌルレクタングルと呼びます）。

また，グラフマネージャでは，レクタングルの座標は必ず，左上，右下の順序に並べる約束になっています。そうなっていない場合は，ヌルレクタングル扱いです。

●リージョン

リージョンは単純な図形では表せないような複雑な形を表現するものです。輪郭が閉じてさえいれば，リージョンの形には制

限がありません。ドーナツ状に穴が空いていても，2つ以上の部分に分割されていてもかまわないのです。

リージョンは通常，レクタングルや楕円の描画ファンクションを利用して作成します。つまり，“以下の描画はリージョンの定義だ”と宣言してから，ラインや図形のフレームを描くSXコールを呼び出すと，図形が描かれる代わりにリージョンの内部データが作成されていくのです。また，2つのリージョン間で差や和，結び，交わりをとることで，単純なリージョンから複雑なリージョンを作り出すこともできます。ただし，リージョンは必ず閉じていなければならないことになっていますので，特に，ラインを使ってリージョンを作成するときには，始点と終点を忘れずに結んでおく必要があります。

なお，リージョンはリロケータブルブロックに置かれ，ハンドルによって参照されます。

●グラフスクリプト

グラフマネージャはグラフスクリプトと呼ばれるデータを扱うこともできます。これはMacintoshでいうpictureに相当し，複数の図形の描画を行う描画コマンド列です。つまり，図形をビットイメージで保持するのではなく，直線やレクタングルなど（を描くコマンド）の組み合わせで表現するものです。

グラフスクリプトの各描画コマンドは内部コードで表されますが，凝ったことをしない限り，内部コードを知る必要はありません。リージョンの作成同様，描画を行い，その描画手順を記録することでグラフスクリプトを作成することができます。

グラフスクリプトもまたリロケータブルブロックに置かれ，ハンドルによって参照されます。

●文字の描画

グラフマネージャはX68000がROMに持っているすべてのフォントを扱うことができます。また，ユーザー定義フォントを使う方法も用意されているようです。

文字描画時には，

・強調
・斜体
・下線
・袋文字（中抜き）
・影つき袋文字

といった，文字飾りを利用することができます。文字飾りは当然，組み合わせて指定できます。反転がありませんが，これは描画モードで，または，描画色と背景色を逆にすることで実現できます。

文字は128×128ドット以内であれば，任意の大きさに縮小・拡大して表示することもできます。ただし，このあたりにはちらほらとバグが残っています。

ウィンドウ

●ウィンドウの各部名称

ウィンドウは大きく分けて，アウトサイドリージョンとインサイドリージョンの2つの部分から構成されます。アウトサイドリージョンはタイトルバーや，輪郭といったウィンドウの外枠部分です。インサイドリージョンは，それ以外の，実際に文字や図形を表示するウィンドウの中身の部分となります。

標準ウィンドウのタイトルバーには，右隅にウィンドウを閉じるためのボタン，クローズボックスがあります。そのほか，ディレクトリを表示するウィンドウでは，タイトルバー上にクリップ，ドライブ名を表示する枠，親ディレクトリに戻る矢印型のボタンなどの，オプションアイテムが並びます。

インサイドリージョンには，ときにスクロールバーや，ウィンドウのサイズを変更するグローボックスが配置されます。とりあえず，これらが特別な領域ではなく，インサイドリージョン中に“置かれているだけ”ということを頭に入れておいてください。なお，細かな話になりますが，スクロールバーはコントロールに，グローボックスはクリップなどと同様，ウィンドウのオプションアイテムに分類されます。前者はコントロールマネージャの，後者はウィンドウマネージャの，それぞれ管理下にあるわけです。

●ウィンドウ構造体

各ウィンドウはそれぞれひとつのウィンドウ構造体を持ち，ウィンドウマネージャはこのウィンドウ構造体によりウィンドウを管理します。ウィンドウ構造体の具体的な構造を図6に示しておきます（簡易マニュアルからの抜粋です）。ウィンドウ構造体はユーザープログラムから直接書き換えるべきものではありませんから，各フィールドの意味を詳しく知っている必要はありませんが，少しだけつまみ食いしてみましょうか。

頭のpixフィールドはウィンドウの描画環境であるグラフ構造体そのものです。ここにグラフ構造体が埋め込まれていることにより，グラフ構造体に対する操作（つまりは描画）をウィンドウ構造体にそのまま適用することができるようになっています。

少し飛んで，wUpdateにはウィンドウ中，再描画しなければならないリージョンへのハンドルが格納されます。ウィンドウを描き直すのはアプリケーション側の仕事です。wUpdateはウィンドウの前後関係が変わったときなどにウィンドウマネージャによって更新されます。ユーザープログラムはアップデートイベント（画面を描き直す必要があるという合図）を受け取ったときにwUpdateで指定されるリージョン内を再描画することになります。

また飛んで，wControlにはウィンドウ上に配置されたコントロールへのハンドルが収められます。こうやって，ウィンドウとコントロールを結びつけるわけです。

その下のwPictureにはグラフスクリプトをぶら下げることができます。このフィールドにグラフスクリプトへのハンドルを格納しておくと，アップデートイベント発生時に，自動的にそのグラフスクリプトを使ってウィンドウの描き直しが行われます。その場合，ウィンドウの持ち主であるタス

図6　ウィンドウ構造体

```
        .offset 0
pix:            ds.b    graph   * グラフマンの描画環境ストラクチャ
wKind:          ds.w    1       * ウィンドウの種類
wVisible:       ds.b    1       * 描画状態
wHilite:        ds.b    1       * アクティブ状態
wClose:         ds.b    1       * クローズボックス
wStatus:        ds.b    1       * ウィンドウマン内部で使用
wOption:        ds.w    1       * ウィンドウアイテムの指定
wOutside:       ds.l    1       * ウィンドウの外枠を示す region
wInside:        ds.l    1       * ウィンドウの内側を示す region
wUpdate:        ds.l    1       * ウィンドウの書き直しを示す region
wDef:           ds.l    1       * ウィンドウ定義関数への handle
wDefData:       ds.l    1       * ウィンドウ定義関数のデータ
wTitle:         ds.l    1       * ウィンドウタイトルへの handle
wTWidth:        ds.w    1       * ウィンドウタイトルの幅
wControl:       ds.l    1       * コントロールへの handle
wNext:          ds.l    1       * 次のウィンドウへの pointer
wPicture:       ds.l    1       * 自動アップデート用 graph script
wTask:          ds.l    1       * ＳＸシェル用タスクＩＤ
window:
```

クには，アップデートイベントは報告されません。この機能は，決まりきった内容しか表示しないウィンドウでは有効な手抜き技となるでしょう。しかし，です。残念ながらSX-WINDOW Ver.1.00～1.02では，この機能は死んでいます（一度でも試してみれば，発見できるバグだと思うのですが）。

●ウィンドウ定義関数と定義ID

SX-WINDOWを使っていると，何種類ものウィンドウがあるのがわかります。ふつうのウィンドウに混じって，タイトルバーしかないような時計.Xのウィンドウとか，電卓.Xのウィンドウ，また，ダイアログにも何通りかがありますね。

ウィンドウの種類はウィンドウオープン時にウィンドウ定義IDで指定します。ウィンドウ定義IDは，ウィンドウ定義関数と密接な関係があります。先ほども簡単に触れましたが，ウィンドウ定義関数はウィンドウの制御や枠の描画を行うモジュールです。ウィンドウの種類ごとに，リソースタイプWDEFを持つリソースとして用意されています。表2に現在SX-WINDOWで使えるWDEFの一覧を示しておきましょう。

さて，ウィンドウ定義IDは，WDEFのリソースID×16＋4ビットのオプションコードの16ビットデータです。標準ウィンドウの場合，オプションコードは1ビットごとに意味を持ち，グローボックスなどのオプションアイテムの有無やON/OFFを指定します。本当はオプションコードは16ビットなのですが，ウィンドウ定義IDでは，その下位4ビットのみを指定します。残りはオープン後，ユーザーアプリケーション側でウィンドウ構造体のwOptionフィールドに格納することになっています（wOptionを設定するSXコールはありません）。なんか奇妙な仕様です。

SXシェル上のプログラム

●タスク

SXシェル上（正確にはタスクマネージャ上というべきでしょうか）では，各ウィンドウ1個1個がタスクとなります。ディレクトリ内容を表示しているウィンドウも，デスクアクセサリ類もそれぞれがひとつの完結したタスクです。とはいっても，ピンボールのように複数のウィンドウを開くタスクもあれば，逆に，ウィンドウを持たないタスクも考えられますから，ウィンドウ＝タスクではありません（なんか変な説明）。そして，SXシェルもまた，タスクのひとつです。

各タスクにはそれぞれタスク管理テーブルが用意されており，タスクマネージャはこれによりタスクを管理します。各タスクはタスクIDと呼ばれる値で識別されます。タスクIDはタスク管理テーブル，そして，ウィンドウ構造体中のwTaskフィールドに収められ，タスクとウィンドウを結びつけています。特にタスクIDが0であるタスクはスーパーユーザーと呼ばれ，タスクマネージャを初期化したりといった雑用を負わされます。通常はSXシェルがスーパーユーザーとなりますが，COMMAND.Xから直接起動しても動くデスクアクセサリ，アプリケーションは，ときにスーパーユーザーとなりうるわけです。

タスクはタスクマネージャのSXコールTSFockなどによって生成され，TSExitにより消滅します。なお，“タスクの生成”と“プログラムの起動”は，概念上はまったく別物ですが，少なくともSX-WINDOWシステムでは，ほぼ同時に起こります。“タスクの消滅”と“プログラムの実行終了”も同様です。

●モジュールヘッダ

SXシェルの上でタスクとして動作するプログラムは，かならずモジュールヘッダと呼ばれるデータブロックを持たなければなりません。その構造は図7のとおりです。モジュールヘッダはRファイルではファイルの先頭，Xファイルでは.endで指定したプログラムの実行開始アドレスの直前に置く約束になっています。

モジュールヘッダの先頭，モジュールタイプは，プログラムの性質（？）を表します。モジュールタイプには，

・リエントラントなプログラム
・リエントラントではないプログラム
・同時に複数実行できないプログラム

の3種類があり，ソース中ではSXCALL.MAC中で定義されたObjectR，ObjectC，ObjectOという記号定数でそれぞれのタイプを表現します。リエントラントとはなにかは，ひとまず横に置いておいて，先に進みましょう。

モジュールサイズは，プログラムのコードサイズです（定数データなども含みます）。この値はR形式のファイルでのみ意味を持ちます。

次のスタートアドレスオフセットは，モジュールヘッダの先頭アドレスから，“TSFockによりプログラムが起動されるときの実行開始アドレス”までのバイト数です。

そして，データサイズが続きます。プログラム起動時には，ここに入れておいたバイト数分の作業用メモリがノンリロケータブルブロックとしてヒープ上に確保され，プログラムには，そこへのポインタがa1レジスタで渡されます。システムがワークエリアを自動的に確保してくれるわけです。

●リエントラントなプログラム

リエントラント（re-entrant：再入可能）というのは，“ルーチンが複数の場所から非同期的に呼び出されても正常に動く”ことを表現する言葉です。サブルーチンの場合でいうと，“サブルーチンからリターンする前に，同じサブルーチンがどこかからさらに呼び出されても大丈夫”であることが保証されていれば，リエントラントといいます。たとえば，再帰する関数・サブルーチンは，リターン前に自分で自分のことを呼び出す形になりますから，リエントラントであることが要求されます。

表2　ウィンドウ定義関数(WDEF)

ID	用途
ID 16	時計.X用
ID 32	標準ウィンドウ
ID 36	ダイアログボックス（影つき）
ID 38	ダイアログボックス（プレーン）
ID 39	ダイアログボックス
ID 40	ダイアログボックス
ID 48	標準ウィンドウ（グラフィック使用可）
ID 50	標準ウィンドウ（グラフィック使用可，クローズボックスのみ）
ID 64	電卓.X用

図7　モジュールヘッダ

```
.offset 0
mdType:         ds.l    1       * モジュールタイプ
mdSize:         ds.l    1       * モジュールサイズ
mdStOffSet:     ds.l    1       * スタートアドレスオフセット
mdDataSize:     ds.l    1       * データサイズ
mdRsv:          ds.l    4       * システム予約
mdXentry:                       * Xファイル時のエントリー(.end で指定する)
mdHead:
```

プログラムをリエントラントにするためには，最低限，複数の呼び出しが同じワークを書き換えあって競合する事態を避けなければなりません。ワークエリアはローカルなものでなければならないのです。具体的には，絶対アドレスでメモリを書き換えることを避け，ワークはスタック上かヒープ上に確保します（定数データは絶対アドレスで参照してもかまいません）。

さて，SXシェル上のプログラムに目を向けましょう。ディレクトリ表示ウィンドウを例に挙げるまでもなく，ときには同じプログラムを複数動作させる場合・させたい場合があります。このとき，プログラムをタスクの数だけメモリ上に置くのは明らかにヒープの無駄遣いです。ワークエリアだけ，タスクごとに用意してやれば，プログラムの実行コード自体は共有できそうに見えませんか。プログラムがリエントラントに作られていれば，それが可能になるのです。

モジュールタイプがOBJECTRのプログラムを起動するとき，タスクマネージャはすでに同じプログラムがメモリ上に存在するかどうかを調べます。もし，存在しなければプログラムを読み込み，同時にモジュールヘッダで示されたデータエリアを確保します。存在するのであれば，データエリアだけを確保して，プログラム本体は読み込みません。プログラム本体の分だけメモリが節約できることになります。同時に複数動かすプログラムは，なるべくリエントラントに作りたいものです。

なお，タスクマネージャはモジュールタイプにのみ頼ってプログラムがリエントラントかどうかを判断します。実際にはリエントラントでないプログラムにObjectRのモジュールタイプを与えた場合の動作は保証されません。もっとも，これを逆手にとれば，ワークの一部を複数のプログラムから共用することもできます。

●タスクマネージャからのイベント取得

アプリケーションプログラムはメインループの中で，必ず1度タスクマネージャにイベントを要求します。イベントを要求するには，イベントに関する情報を格納するイベントレコードへのポインタと，イベントマスクをスタックに積んで，SXコールTSEventAvailを呼び出します（このときタスクの切り替えが行われますが，アプリケーションは気づきません）。イベントマスクは，自分にとって不要なイベントに煩わされないようマスクするものです。TSEventAvailからは，マスクしなかったイベントのうち，最も優先順位の高いイベントが，イベントコードと付随情報の形でイベントレコードに返されます。

タスクマネージャから得られるイベントには以下のものがあります。

・ヌルイベント
・マウス左ボタンダウンイベント
・マウス左ボタンアップイベント
・マウス右ボタンダウンイベント
・マウス右ボタンアップイベント
・キーダウンイベント
・キーアップイベント
・アップデートイベント
・アクティベートイベント
・システムイベント1
・システムイベント2
・ユーザーイベント1
・ユーザーイベント2

アクティベートイベントまではイベントマネージャの管轄です。タスクマネージャは，イベントマネージャから送られたイベントもまとめてアプリケーションに返します。

ヌルイベントは，何もイベントがないときに定期的に発行されるイベントです。疑似マルチタスクの鍵でしたね。

マウスボタンとキーのアップ・ダウンイベントについては触れるまでもないでしょう。それぞれ，ボタン・キーが押されたとき，離されたときに発生するイベントです。

アップデートイベントはさっきも顔を出しました。ウィンドウの前後関係が変化したなどの理由により，いままで見えなかった部分が見えるようになったから，ウィンドウをアップデート（再描画）しなさい，というイベントです。ウィンドウの枠（アウトサイドリージョン）の再描画はウィンドウマネージャ（というかウィンドウ定義関数）が行いますから，ユーザープログラム側では，ウィンドウの中身（インサイドリージョン）のみを描き直すことになります。なお，アップデートイベントは非常に優先順位の低いイベントです。ほかの重要なイベントがない場合にのみ送られてきます。これには，画面の再描画をなるべくまとめて行うことで，全体の処理効率を上げる意味があります。

アクティベートイベントは，ウィンドウのアクティブ状態が変化したときに発生するイベントです。どのウィンドウがアクティブになったかという情報が同時に与えられますので，それが自分のウィンドウであれば自分をアクティブ状態にし（それを覚えておく），そうでなければ非アクティブ状態にすることになります。アクティベートイベントは優先順位の高いイベントで，ウィンドウの状態が変化したときには，マウスボタンの押し下げやキー入力がイベントキューに残っていたとしても，即座に発行されます。

システムイベント1，2はタスクマネージャからのイベントです。具体的な内容はイベントレコードにタスクマネージャのイベントコードとして返されます。システム1は全タスクに対するイベント，システム2は各タスクに個別に送られるイベントということになっていますが，中身は同じです。タスクマネージャからのイベントには，タスクを終了せよという指示など，各プログラムが必ずサポートしなければならない重要なものが含まれます。

また，タスクマネージャのイベントコード中128以降はアプリケーションに解放されています。これらのコードに適当な意味を持たせ，アプリケーション側からイベントを発行することでタスク間通信が行えます。

ユーザーイベントはユーザー用です。いまのところは使われていません（きっと）。

サンプルプログラム

まだまだ説明が不十分な気もしますが，そろそろプログラミングに進み，細かな部分はサンプルプログラムを見ながら解説したいと思います。

リスト1はウィンドウを開き，文字列を適当に表示するプログラムです。実行ファイルは，

```
A>AS WINTEST
A>LK WINTEST
```

により作成します。付録ディスク中のインクルードファイルをがりがり取り込んでいますから，AS Ver.1.0xを使っている人はカレントディレクトリに，Ver.2.0を使っている人は環境変数includeで指定したディレクトリに，各インクルードファイルを置いておいてください。

では，プログラム各部の意味・動作を説明します。

●マクロ定義（32～41行）

2つのマクロを定義しています。ひとつ目のSXCALLはSXコールを呼び出すだけの簡単なマクロです。べつに.dc.wでコール番号を埋め込んでも，DOSCALL.MACで定義されているマクロDOSを使うことにしてもよかったのですが，ほんの気分であります。

2つ目のLASCIIは，簡易マニュアルでLASCII文字列と呼ばれている形式の文字列データを定義するマクロです。LASCIIは文字列の頭に1バイトで文字列長を置く形式で，LASCII形式の文字列を.dc.bで用意するときには，自分で文字列の長さを数えて先頭に置く必要があります。それが嫌だったので，マクロにして，文字列の長さはアセンブラに計算させるようにしてみました。

●データエリアの定義（45〜53行）

データエリアの大きさと，その内部構造を.offset疑似命令で定義しています。起動時に確保され，a1レジスタで渡されるデータエリアに対するアクセスは，ここで定義したオフセットを使い，ディスプレイスメントつきアドレスレジスタ間接形式で行います。

WINTEST.Xでは，本当に最小限のワークしか用意していません。

winptrにはウィンドウをオープンしたときに返されるウィンドウ構造体へのポインタを格納します。以降，ウィンドウを指定する必要があるときには，このポインタを使います。また，リストではwinptrをウィンドウがオープンされているかどうかのフラグに兼用しています。winptrが0ならウィンドウは未オープンです。

winsizeはウィンドウオープン時にウィンドウの位置と大きさを指定するのに使うレクタングル構造体です。

eventrecはイベント取得時に使うイベントレコードで，その先頭アドレスをTSEventAvailに渡すことになります。

activeflagは自分が現在アクティブウィンドウかどうか（最前面に表示されているかどうか）を覚えておく1バイトのフラグです。非0ならアクティブと決めます。

それ以降はスタック領域です。すっかり話し忘れていましたが，データエリアが確保された際に，sp（usp）はデータエリア末尾を指すようタスクマネージャによって初期化されます。データエリアにはその分も用意しておかなければなりません。

●モジュールヘッダ（60〜65行）

このプログラムはリエントラントです。以下，コードサイズ，スタートアドレスオフセットはラベルの差で，データエリアの大きさは.offsetで定義した記号定数で指定しています。

●exec時の処理（71〜133行）

COMMAND.Xから直接起動した場合には，71行から実行が始まります。このアドレスは528行の.end疑似命令で指定してあります。多少細工してあり，アセンブル時に，

A>AS /sNDEBUG WINTEST

のようにシンボルNDEBUGを定義すると，75〜81行の部分がアセンブルされ，さもなくば，85〜131行がアセンブルされるようになっています。NDEBUGを定義した場合には，COMMAND.X上からは起動できない旨のメッセージを出してWINTEST.Xは終了します。NDEBUGが定義されていない場合は，デバッグモードということで，デバッグ用のシェルSXWDB.Xを起動し，WINTEST.Xはその上で走ります（SXWDB.Xは開発キット中に含まれます）。SXWDB.Xはアプリケーションに代わって，スーパーユーザーの役割を果たしてくれます。

85行以下，SXWDB.Xを起動するところまでは，子プロセスの生成の常套手段です。スタックポインタを初期化してから，DOSコールsetblockで不要なメモリを切り離し，DOSコールexecでSXWDB.Xを起動します。SXWDB.Xはpathの通ったディレクトリに置いておくようにしてください。

SXWDB.Xは起動後，WINTEST.XをTSFockを使って再起動します。そして，制御はWINTEST.Xに戻り，142行から再スタートします。

●TSFock時の処理（142〜154行）

TSFockにより起動されたプログラムにはレジスタを介していくつかの情報が渡されます。リスト1で使っているのは，

a1　データエリアへのポインタ
a2　コマンドライン引数
d0　自分のタスクID

です。

a1は以降のデータエリアアクセス時のベースアドレスとなりますから，適当なアドレスレジスタ（リスト1ではa5）に転送しておきます。

続いて，アクティブかどうかのフラグと，ウィンドウ構造体へのポインタを0で初期化しています。データエリアの初期値は不定ですから，必要であれば，自分で初期化しなければなりません。

その後，d0が負かどうかを調べています。d0が負の場合は，起動途中でエラーが発生したことを意味しています。WINTEST.Xはエラー時にはそのまま終了処理に移ります。

●初期化（472〜520行）

サブルーチンinitでは，ウィンドウの表示を行います。

まず，SXコールTSTakeParamを使って，a2レジスタで渡されたコマンドライン引数からウィンドウ表示座標を取り出します。SXシェル上で“画面状態を保存”したとき保存されたウィンドウの位置が，

－w nnn, nnn, nnn, nnn

の形式で与えられる場合があるからです（SYSDTOP.SXをダンプして納得してください）。この指定があった場合，TSTakeParamはd0の最下位ビットを1にして戻ります。このビットが1であれば，winsizeにはウィンドウの位置が格納されていますから，ウィンドウのオープン処理に移りま

リスト1

```
  1: *
  2: *          ウィンドウを開いて文字を表示する
  3: *
  4:            .include        doscall.mac
  5:            .include        sxcall.mac
  6:            .include        sxcall.equ
  7: *
  8: TRUE               equ     -1              *真
  9: FALSE              equ     0               *偽
 10: *
 11: CR                 equ     $0d             *復帰コード
 12: LF                 equ     $0a             *改行コード
 13: *
 14: WDEFID             equ     WI_STD          *ウィンドウ定義関数ID
 15:                                            *（標準ウィンドウ）
 16: WINOPT             equ     WC_GBOX|WC_GBOXON
 17:                                            *ウィンドウオプション
 18:                                            *（グローボックスあり&ON）
 19: WINOPTLOW          equ     WINOPT&%1111    *ウィンドウオプション下位4ビット
 20: WINDEFID           equ     WDEFID<<4|WINOPTLOW
 21:                                            *ウィンドウ定義ID
 22: *
 23: WINH               equ     208             *ウィンドウ横ドット数
 24: WINV               equ     160             *ウィンドウ縦ドット数
 25: WINSIZE            equ     WINH<<16|WINV   *ウィンドウの大きさ（point風）
 26: *
 27: EVENTMASK          equ     EM_MSLDOWN|EM_UPDATE|EM_ACTIVATE|EM_SYSTEM1|EM_SY
STEM2
 28:                                            *イベントマスク
 29: *
 30: G_PLAIN            equ     0               *プレーンな書体
 31: *
 32: SXCALL  macro      callno                  *SXコールマクロ（ほんの気分）
 33:         .dc.w      callno                  *
```

す。

位置の指定がなかった場合は，TSGetWindowPosを使って，タスクマネージャに適当な位置を決めてもらいます。TSGetWindowPosはd0に適当な座標（上位ワードにx座標，下位ワードにy座標）を返しますから，この座標をウィンドウの左上隅の座標とし，ウィンドウの大きさを加えて右下隅の座標を求め，それぞれwinsizeに格納します。そして，492行に合流して，ウィンドウをオープンする態勢に入ります。

492行では，まず自分のタスクIDを得ています。これはウィンドウをオープンするときに必要な情報です。起動時のd0にもタスクIDは格納されていたわけですが，ごちゃごちゃやっている間にd0は破壊されていますから，TSGetIDで取得しなおしています。

以下，ウィンドウをオープンするSXコールWMOpenに与える引数をどんどんスタックに積んでいきます。順に，

・タスクID
・クローズボックスの有無（非0ならあり）
・ウィンドウをどのウィンドウの後ろに置くかの指定（−1を指定すると一番手前）
・ウィンドウ定義ID（標準ウィンドウで，グローボックスあり）
・ウィンドウを表示するかどうか（非0なら表示）
・ウィンドウのタイトル（LASCIIへのポインタ）
・ウィンドウの位置（レクタングル構造体へのポインタ）
・ウィンドウ構造体を格納するメモリアドレス（0ならメモリマネージャに確保してもらう）

です。ウィンドウ定義IDは14〜20行のあたりで記号定数に定義してあります。14行のシンボルWDEFIDの値を変えればいろいろなタイプのウィンドウを表示することができますし，WINOPTの値を変更してもそれなりに遊べます。気が向いたら試してみてください。

WMOpenから戻ってきた時点でNフラグが立っているときはエラーです。正常にオープンできたときにはa0にウィンドウ構造体へのポインタが格納されていますから，ワークにしまっておきます。

それから，510行で，ウィンドウオプションをウィンドウ構造体に直接書き込んでいます。WMOpenを呼んだときに“グローボックスあり”までは設定できているのですが，グローボックスはONに設定しないと，

```
 34:            endm                                        *
 35: *
 36: LASCII     macro    string                             *LASCII文字列を定義するマクロ
 37:            local    tos,eos                            *
 38:            .dc.b    eos-tos                            *
 39: tos:       .dc.b    string                             *
 40: eos:                                                   *
 41:            endm                                        *
 42: *
 43: *          データエリアの定義
 44: *
 45:            .offset 0
 46: *
 47: winptr:             .ds.l    1                         *ウィンドウ構造体へのポインタ
 48: winsize:            .ds.b    rect                      *ウィンドウの位置
 49: eventrec:           .ds.b    tsevnt                    *タスクマネージャ用のイベントレコード
 50: activeflag:         .ds.b    1                         *自分がアクティブかどうかのフラグ
 51:                     .even
 52:                     .ds.l    256                       *スタック
 53: sizeof_WORK:                                           *データエリアの総バイト数
 54: *
 55:            .text
 56:            .even
 57: *
 58: *          モジュールヘッダ
 59: *
 60: head:
 61:            .dc.l    OBJECTR                            *リエントラント
 62:            .dc.l·   tail-head                          *実行コードのバイト数
 63:            .dc.l    entry-head                         *スタートアドレスオフセット
 64:            .dc.l    sizeof_WORK                        *データエリアのバイト数
 65:            .dc.l    0,0,0,0                            *システム予約
 66: *
 67: *          EXEC時のエントリ
 68: *          COMMAND.Xから起動したときはここから始まる
 69: *          (そのように.endで指定しておく)
 70: *
 71: exec_entry:
 72:
 73: .ifdef     NDEBUG
 74:
 75:            pea.l    sorrymes(pc)
 76:            DOS      _PRINT
 77:            DOS      _EXIT
 78:
 79: sorrymes:
 80:            .dc.b    'SXシェル上で起動してください',CR,LF,0
 81:            .even
 82:
 83: .else
 84:
 85:            lea.l    inisp(pc),sp                       *sp初期化
 86:
 87:            lea.l    16(a0),a0                          *余分なメモリを切り離す
 88:            sub.l    a0,a1                              *
 89:            move.l   a1,-(sp)                           *
 90:            move.l   a0,-(sp)                           *
 91:            DOS      _SETBLOCK                          *
 92:
 93:            clr.l    -(sp)                              *SXカーネルをpathから検索し
 94:            pea.l    cmdarg(pc)                         *  起動する
 95:            pea.l    fname(pc)                          *(このプログラム自体は
 96:            move.w   #2,-(sp)                           *  SXカーネルから再起動される)
 97:            DOS      _EXEC                              *検索
 98:            tst.l    d0                                 *SXカーネルを見つけた?
 99:            bmi      error                              *  見つからなかった
100:            clr.w    (sp)                               *
101:            DOS      _EXEC                              *起動
102:            tst.l    d0                                 *SXカーネルを起動できた?
103:            bmi      error                              *  起動できなかった
104:
105:            DOS      _EXIT                              *正常終了
106: *
107: error:     pea.l    errmes(pc)                         *SXカーネルが起動できなかった
108:            DOS      _PRINT                             *
109:            move.w   #1,-(sp)                           *エラー終了
110:            DOS      _EXIT2                             *
111:
112: *
113: *          EXEC時用データ/データ
114: *          SXシェル上から起動したときはここから始まる
115: *
116:            .data
117:            .even
118: *                    123456789012
119: fname:     .dc.b    'SXWDB -D -K',0                    *SXカーネルを起動するコマンド行
120:                                                        *(とりあえずデバッグ用のシェル)
121: fnamee:    .ds.b    100-12                             *(フルパスで上書きされるから
122:                                                        *  90数バイト必要)
123: errmes:    .dc.b    'SXカーネルが起動できませんでした',CR,LF,0
124:                                                        *エラーメッセージ
125: *
126:            .bss
127:            .even
128: *
129: cmdarg:    .ds.b    256                                *コマンド行引数格納用
130:            .ds.b    64                                 *初期スタック
131: inisp:
132:
133: .endif
```

見慣れたあの形では表示されないのです(510行を削って確かめてみましょう)。

そして，GMSetGraphで自分のウィンドウ（に対応したグラフ）をカレントグラフにしてから，WMDrawGBoxでグローボックスを描きます。グローボックスはインサイドリージョンに置かれますから，表示・再表示はアプリケーション側の仕事です。

これでウィンドウの表示が終わりました。描かれるのは“スクロールバーなしでグローボックスだけがある”という多少邪道なウィンドウです。なお，ウィンドウの中身は，あとでアップデートイベントが発生したときに描きます。

●メインループ（158〜175行）

ずーっと戻って，メインループです。イベントレコードとイベントマスクをTSEventAvailに渡して，イベントを得ます。イベントマスクは27行で記号定数EVENTMASKに定義してあります。

- ・マウス左ボタンダウン
- ・アップデート
- ・アクティベート
- ・システム1，2

という必要最小限のイベントのみを受け取るようにしています。

TSEventAvailから戻った時点で戻り値のd0がハードウェアエラーによるアボートを示していた場合は終了します。

エラーがなければ，イベントレコードのtnWhatフィールドにイベントコードが格納されています。そこで，ジャンプテーブルにより，イベントに対応した処理ルーチンをサブルーチンコールします。

処理がすんで戻ってきたら，また，ループの頭に飛んで，次のイベントを要求します。これがえんえんと続くのです。

●マウス左ボタンダウンイベント発生時の処理（208〜244行）

208〜244行がマウスの左ボタンが押された場合の処理です。イベントレコードのtnWhereフィールドには，どのウィンドウ上でマウスボタンが押されたかが，ウィンドウ構造体へのポインタの形で格納されていますから，まず，自分のウィンドウかどうかを調べます。違えば関係ありませんから即戻ります。

自分のウィンドウ上だということがわかったら，現在，自分がアクティブウィンドウかどうか，activeflagを調べてみます。アクティブウィンドウかどうかで，処理が多少違うのです。ソース上の順序と逆ですが，先にアクティブウィンドウだった場合の処理を見てもらいましょう。231行からです。

```
134:
135: *
136:          .text
137: *
138: *        TSFock時のエントリ
139: *
140: WORK     reg     a5                          *データエリアのベースアドレス
141: *
142: entry:
143:          movea.l a1,WORK                     *WORK=a5
144:                                              *以下データエリアはa5からの
145:                                              *  オフセットでアクセスする
146:          sf.b    activeflag(WORK)            *アクティブフラグをクリア
147:          clr.l   winptr(WORK)                *ウィンドウ構造体へのポインタを
148:                                              *  クリア（ウィンドウ未オープン）
149:
150:          tst.l   d0                          *d0が負なら
151:          bmi     exit                        *  起動途中でエラーが発生した
152:
153:          bsr     init                        *初期化する
154:          bmi     exit                        *N=1なら初期化に失敗した
155: *
156: *        メインループ
157: *
158: mainloop:
159:                                      *イベントを取得
160:          pea.l   eventrec(WORK)              *イベント格納用レコード
161:          move.w  #EVENTMASK,-(sp)            *イベントマスク
162:          SXCALL  __TSEventAvail              *イベントを得る
163:          addq.l  #6,sp                       *（イベントキューからは取り除かない）
164:          cmpi.w  #ER_ABORT,d0                *エラー？
165:          beq     exit                        *  そうなら即終了
166:
167:                                      *イベントに応じた処理ルーチンに分岐する
168:          move.w  eventrec+tnWhat(WORK),d0
169:                                              *d0=イベントコード
170:          andi.w  #$000f,d0                   *念のためマスク
171:          add.w   d0,d0                       *分岐先への
172:          move.w  ejmptbl(pc,d0.w),d0         *  相対距離を求める
173:          jsr     ejmptbl(pc,d0.w)            *対応ルーチンを呼び出す
174:
175:          bra     mainloop                    *えんえんと繰り返す
176:
177: *
178: *        イベントに対応した処理ルーチンへのジャンプテーブル
179: *
180: ejmptbl:
181: x = ejmptbl
182:          .dc.w   noop-x                      *ヌル
183:          .dc.w   mouseLdown_event-x          *マウス左ボタンダウン
184:          .dc.w   noop-x                      *マウス左ボタンアップ
185:          .dc.w   noop-x                      *マウス右ボタンダウン
186:          .dc.w   noop-x                      *マウス右ボタンアップ
187:          .dc.w   noop-x                      *キーダウン
188:          .dc.w   noop-x                      *キーアップ
189:          .dc.w   update_event-x              *アップデート
190:          .dc.w   noop-x                      *        （システム予約）
191:          .dc.w   activate_event-x            *アクティベート
192:          .dc.w   noop-x                      *        （システム予約）
193:          .dc.w   noop-x                      *        （システム予約）
194:          .dc.w   system1_event-x             *システム1
195:          .dc.w   system2_event-x             *システム2
196:          .dc.w   noop-x                      *ユーザー1
197:          .dc.w   noop-x                      *ユーザー2
198:
199: *
200: *        自分に関係ないイベントの処理
201: *        （マスクしているから，本当はここにはこないが念のため）
202: noop:
203:          rts
204:
205: *
206: *        マウス左ボタンダウンイベントの処理
207: *
208: mouseLdown_event:
209:                                      *自分のウィンドウの上かどうか調べる
210:          move.l  winptr(WORK),a1             *ウィンドウ構造体へのポインタ
211:          cmp.l   eventrec+tnWhom(WORK),a1
212:                                              *自分かな？
213:          bne     LDretn                      *違った
214:                                      *自分のウィンドウの上だった
215:          tst.b   activeflag(WORK)            *いまアクティブだっけ？
216:          bne     callWM                      *前からアクティブだった
217:                                      *アクティブにする
218:          move.l  a1,-(sp)                    *自分のウィンドウを
219:          SXCALL  __WMSelect                  *  一番上にする
220:          addq.l  #4,sp                       *
221:                                      *タイトルバー上かどうか調べる
222:          move.l  eventrec+tnWhom2(WORK),-(sp)
223:                                              *
224:          SXCALL  __WMFind                    *どこでボタンが押されたか？
225:          addq.l  #4,sp                       *
226:          cmpi.w  #W_INDRAG,d0                *タイトルバーの上？
227:          bne     LDskip                      *  違った
228:          SXCALL  __EMLStill                  *まだボタンは押されている？
229:          beq     LDskip                      *  もう離された
230:
231: callWM:  pea.l   eventrec(WORK)              *あとはウィンドウマネージャに
232:          move.l  a1,-(sp)                    *  まかせてしまう
233:          SXCALL    SXCallWindM               *
```

本来，ここでやらなければならないことはたくさんあります。まず，WMFindを使って，ウィンドウのどの位置でマウスボタンが押されたか調べます。結果はパートコードというヤツで返されますから，それに応じて処理を振り分けます。

タイトルバー上だったらマウスボタンが離されるまでのあいだマウスの動きを追跡しつつ，点線で描いたウィンドウの枠を動かし，ボタンが離されたその位置にウィンドウを移動します。

クローズボックス上であれば，マウスカーソルがクローズボタン上にあるあいだ，クローズボックスをハイライト表示し，ボタンが離されたときに，まだマウスカーソルがクローズボックス内であれば，ウィンドウを閉じます。そうでなけれな，何もしてはいけません。ユーザーは途中で気が変わったのです。

グローボックスがドラッグされた場合は，やはりマウスカーソルを追跡し，ボタンが離されたら，ウィンドウの大きさを変更します。また，グローボックスがダブルクリックされた場合はズームですから，ウィンドウを最大に拡大（または元に戻す）します。

とても面倒ですね。Macintoshでは実際にこういうことをやっているようです。SX-WINDOWにも該当コールがありますから，同じことをやろうと思えばできます。でも，タスクマネージャに任かせてしまうという技があるのでした。

リスト1の233行で呼び出しているSXCallWMが上に挙げた一切の処理を受け持ちます。イベントレコード（その中にボタンが押された座標を収めたフィールドがあります）と，ウィンドウ構造体へのポインタを渡すだけという簡便さです。最終的にSXCallWMは，マウスのボタンが離された位置のパートコードを返します。それがクローズボックスを表していたら，終了処理に飛びます。

では，非アクティブだった場合に戻りましょう。224行からです。まず，WMSelectにより，自分を最前面に出し，アクティブウィンドウにします。それから，WMFindにより，ボタンが押された位置のパートコードを得て，それがタイトルバーを意味していた場合にのみ，SXCallWMを呼び出し，ウィンドウのドラッグを任せます。下敷になっていたウィンドウをいきなりドラッグすることを許しているわけです。

マウス左ボタンダウンイベント処理の最後では，TSGetEventにより，いま処理した

```
234:         addq.l  #8,sp                       *
235:
236:         cmpi.w  #W_INCLOSE,d0               *クローズされた？
237:         beq     exit                        *  そうなら終了する
238:
239: LDskip: pea.l   eventrec(WORK)              *イベント格納用レコード
240:         move.w  #EVENTMASK,-(sp)            *イベントマスク
241:         SXCALL  __TSGetEvent                *イベントを取り除く
242:         addq.l  #6,sp                       *
243:
244: LDretn: rts                                 *処理完了
245:
246: *
247: *       アップデートイベントの処理
248: *
249: update_event:
250:                                     *自分のウィンドウかどうか調べる
251:         move.l  winptr(WORK),a1             *ウィンドウ構造体へのポインタ
252:         cmp.l   eventrec+tnWhom(WORK),a1
253:                                             *自分かな？
254:         bne     Uretn                       *  違った
255:                                     *自分のウィンドウだった
256:         move.l  a1,-(sp)                    *アップデートリージョンで
257:         SXCALL  __WMUpdate                  *  クリップする
258:
259:         bsr     doupdate                    *画面を描き直す
260:
261:         SXCALL  __WMUpdtOver                *クリップリージョンを元に戻す
262:         SXCALL  __WMDrawGBox                *グローボックスを描く
263:         addq.l  #4,sp                       *
264: *
265: Uretn:  rts                                 *処理完了
266:
267: *
268: *       アクティベートイベントの処理
269: *
270: activate_event:
271:                                     *自分のウィンドウかどうか調べる
272:         move.l  winptr(WORK),d0             *ウィンドウ構造体へのポインタ
273:         beq     Aretn                       *0ならどのウィンドウでもない
274:         cmp.l   eventrec+tnWhom(WORK),d0        *自分かな？
275:         seq.b   activeflag(WORK)            *自分だったらアクティブフラグをON
276:                                             *そうでなければOFF
277: Aretn:  rts                                 *処理完了
278:
279: *
280: *       タスクマネージャからのイベント1，2の処理
281: *
282: system1_event:
283: system2_event:
284:         move.w  eventrec+tnWhat2(WORK),d0
285:                                             *d0=イベントの種類
286:         cmp.w   #ENDTSK,d0                  *タスクの終了？
287:         beq     exit                        *  そうなら終了する
288:         cmp.w   #CLOSEALL,d0                *ウィンドウ全クローズ？
289:         beq     exit                        *  そうなら終了する
290:         cmp.w   #WINDOWSELECT,d0            *ウィンドウのセレクト？
291:         bne     Sretn
292:
293:         move.l  winptr(WORK),-(sp)          *自分のウィンドウを
294:         SXCALL  __WMSelect                  *  一番上にする
295:         addq.l  #4,sp                       *
296:
297: Sretn:  rts                                 *処理完了
298:
299: *
300: *       終了処理
301: *
302: exit:
303:         move.l  winptr(WORK),d0             *d0=ウィンドウ構造体へのポインタ
304:         beq     done                        *0...ウィンドウは未オープン
305:
306:         move.l  d0,-(sp)                    *ウィンドウを破棄する
307:         SXCALL  __WMDispose                 *
308:         addq.l  #4,sp
309:
310: done:   clr.l   -(sp)                       *終了コード
311:         SXCALL  __TSExit                    *タスクを終了する
312:
313: *       ↑
314: *ここまでが基本部分（どのプログラムでもだいたい同じ）
315: ********************************************************************
316: *以下，プログラム固有のアップデートルーチンと初期化ルーチン
317: *       ↓
318:
319: *
320: *       画面を（再）描画する
321: *
322: doupdate:
323:         move.l  a1,-(sp)                    *a1=ウィンドウ構造体へのポインタ
324:                                             *   =そのグラフポートへのポインタ
325:         SXCALL  __GMSetGraph                *自分の描画環境をカレントグラフにする
326:         addq.l  #4,sp                       *
327:
328:         lea.l   testdat0(pc),a1             *テスト用データ
329:         bsr     test1                       *16×16ドットフォントで遊ぶ
330:         bsr     test1                       *12×12～
331:         bsr     test1                       *24×24～
332:
```

マウス左ダウンイベントをイベントキューから取り除きます。TSEventAvailはイベントキューを覗き見るだけで，実際にはイベントを取り除きません。関係のないイベントまで取り出してしまっては，ほかのプログラムが困りますから，自分で処理した場合にのみ，TSGetEventを呼び出します。

●アップデートイベント発生時の処理（249～265行）

アップデートイベント時も，まず，そのイベントが自分のウィンドウに向けられたものかどうかを，ウィンドウ構造体へのポインタの比較により判断します。自分だということになれば，アップデート開始です。

アップデートすべきリージョンはウィンドウ構造体のwUpdateに格納されています。このリージョン内だけを再描画すればよいわけですが，わざわざチェックするのも大変です。でも，そこはそれ，うまい方法がちゃんと用意されています。

WMUpdateを呼び出すと，以降の描画範囲がアップデートリージョンに制限されるのです。そのうえで全画面を描き直すと，実際にはアップデートリージョンだけ描き直されます。アップデートが済んだらWMUpdtOverで，描画範囲の制限を解除しておきます。

ここで，WMUpdtOverのあとでグローボックスを描いていることに注意してください。グローボックスはアップデートリージョン外にあっても描き直さなければならない場合があるので，こういうことをしてみました（もう少しスマートな方法もあるような気はします）。

さて，実際に描画を行うのは322行以下のサブルーチンです。必ず最初に自分のウィンドウ（に対応したグラフ）をカレントグラフにします。カレントグラフは各タスクが勝手に変更しますから，これを怠るとほかのウィンドウに落書きしてしまうことになります。

以下，test1～test4の各サブルーチンでは，文字をいろいろな文字飾りで表示したり，拡大・縮小したりして遊んでいます。ここで使っている各SXコールの機能は単純なものばかりですから，説明は省略します。注釈と簡易マニュアルを参照してください。

なお，描画結果を見ると，ゴミが表示されている部分や文字が欠けている部分が見つかるかもしれませんが，それは私のミスでも読者のミスでもありません。

●アクティベートイベント発生時の処理（270～277行）

```
333:         bsr     test2                              *文字を拡大縮小して遊ぶ
334:         bsr     test3                              *
335:         bsr     test4                              *
336:
337:         rts                                        *描画完了
338:
339: test1:                    *いろいろな文字飾りをつけて文字列を描いてみる
340:         link    a6,#0
341:
342:         move.w  (a1)+,-(sp)                        *フォントの種類をセット
343:         SXCALL  __GMFontKind                       *
344:         move.l  (a1)+,-(sp)                        *フォントのサイズをセット
345:         SXCALL  __GMFontSize                       *
346:
347:         move.l  (a1)+,-(sp)                        *初期ペン座標
348:         move.w  (a1)+,d1                           *d1=改行幅
349:
350:         lea.l   testdat1(pc),a2                    *テスト用データ
351: loop1:  SXCALL  __GMMove                           *ペン座標を移動して
352:         move.w  (a2)+,d0                           *
353:         bmi     test1retn                          *
354:
355:         move.w  d0,-(sp)                           *書体をセットして
356:         SXCALL  __GMFontFace                       *
357:         move.l  (a2)+,-(sp)                        *文字列を描画する
358:         SXCALL  __GMDrawStrZ                       *
359:         addq.l  #6,sp                              *
360:         add.w   d1,ptY(sp)                         *改行
361:         bra     loop1                              *繰り返す
362: test1retn:
363:         unlk    a6
364:         rts
365: *
366: test2:                    *文字を縮小・拡大してみる
367:         link    a6,#0
368:
369:         move.w  #G_PLAIN,-(sp)                     *書体ノーマル
370:         SXCALL  __GMFontFace                       *
371:
372:         move.l  #$0004_017c,-(sp)                  *初期座標(4,380)
373:         SXCALL  __GMMove
374:         move.l  #$0001_0001,d1                     *フォントサイズ増分
375:         move.l  #$0008_0008,-(sp)                  *初期フォントサイズ
376:
377:         moveq.l #24-1,d7
378: loop2:  SXCALL  __GMFontSize                       *フォントサイズをセット
379:
380:         move.w  #'も',-(sp)                        *1文字表示
381:         SXCALL  __GMDrawChar                       *
382:         addq.l  #2,sp                              *
383:
384:         add.l   d1,(sp)                            *フォントサイズを増す
385:         dbra    d7,loop2
386:
387:         unlk    a6
388:         rts
389: *
390: test3:                    *文字を縮小・拡大してみる(その2)
391:         link    a6,#0
392:
393:         move.l  #$0040_0008,d1                     *初期フォントサイズ
394:         move.l  #$0190_0004,-(sp)                  *初期座標(400,4)
395:
396:         moveq.l #20-1,d7
397: loop3:  SXCALL  __GMMove                           *ペン位置をセット
398:
399:         move.l  d1,-(sp)                           *フォントサイズをセット
400:         SXCALL  __GMFontSize                       *
401:         move.w  #'み',-(sp)                        *1文字表示
402:         SXCALL  __GMDrawChar                       *
403:         addq.l  #6,sp                              *
404:
405:         addq.w  #1,d1                              *フォントサイズを増す
406:         add.w   d1,ptY(sp)                         *表示座標をずらす
407:         dbra    d7,loop3
408:
409:         unlk    a6
410:         rts
411: *
412: test4:                    *最大サイズで1文字表示してみる
413:         link    a6,#0
414:
415:         move.l  #$0080_0080,-(sp)                  *フォントサイズ128×128
416:         SXCALL  __GMFontSize                       *
417:         move.l  #$00dc_00a0,-(sp)                  *座標(220,160)
418:         SXCALL  __GMMove                           *
419:         move.w  #'髫',-(sp)                        *1文字表示
420:         SXCALL  __GMDrawChar                       *
421:
422:         unlk    a6
423:         rts
424: *
425: testdat0:
426:         .dc.w   G_ROM16
427:         .dc.w   16,16
428:         .dc.w   4,4
429:         .dc.w   20
430:
431:         .dc.w   G_ROM12
```

アクティベートイベント発生時にやるべきことは単純です。イベントレコードに格納された“いまアクティブになるべきウィンドウ”が自分であれば，内部のフラグをONに，そうでなければOFFにするだけです。フラグの操作はseq.b一発ですね。

●システムイベント発生時の処理（282〜297行）

システムイベント1と2の処理は，兼用しています。それでかまわないようです。

WINTEST.Xでは，タスクマネージャからのイベントのうち，本当に最小限のもののみをサポートしています。

・タスクを終了しろというイベント
・ウィンドウを閉じろというイベント
・アクティブウィンドウになれというイベント

の3つです。上2つはそのまま終了処理になだれ込めばおしまいです。また，3番目のヤツは，WMSelectで自分をアクティブにすればよいわけです。

●終了時処理（302〜311行）

終了時には，ウィンドウを閉じ，自分で確保したすべてのメモリを解放しなければなりません。

WINTEST.Xでは明示的に確保したメモリはありませんが，WMOpenでウィンドウを開いたときにウィンドウ構造体格納用のメモリを確保していますから，これを解放します。ウィンドウを閉じる処理とメモリを解放する処理を別々に行うこともできますが，ここでは，それらを同時にやってくれるWMDisposeを使っています。

そして，TSExitで終了です。簡易マニュアルによればTSExitで終了コードを返すとあるのでそれに従っています。しかし，SX-WINDOW標準のデスクアクセサリのなかには終了コードを返していないものもあるようですから，どれほどの意味があるのかはわかりません。

＊

以上，SX-WINDOW上のプログラミングのほんのさわりの部分でした。次号ではC言語で開発する場合についてなども解説する予定です。では，また。

≪参考文献≫

1) Apple Computer, Inside Macintosh Volume I〜V（日本語版），Addison-Wisley
2) スティーブン・チェルニコフ，マッキントッシュの道具箱 Vol.I〜II，パーソナルメディア
3) スコット・ナスター，続マッキントッシュの道具箱，パーソナルメディア
4) BNN第二企画部，インサイドマック徹底ガイド 上下巻，BNN
5) 川又ほか，Macintoshの「技術」徹底解剖，インターフェース1988年12月号，CQ出版


```
432:            .dc.w   12,12
433:            .dc.w   220,4
434:            .dc.w   20
435:
436:            .dc.w   G_ROM24
437:            .dc.w   24,24
438:            .dc.w   4,160
439:            .dc.w   28
440: *
441: testdat1:
442:            .dc.w   G_PLAIN
443:            .dc.l   mes0
444:            .dc.w   G_BOLD
445:            .dc.l   mes1
446:            .dc.w   G_ITALIC
447:            .dc.l   mes2
448:            .dc.w   G_ULINE
449:            .dc.l   mes3
450:            .dc.w   G_OLINE
451:            .dc.l   mes4
452:            .dc.w   G_SHADOW
453:            .dc.l   mes5
454:            .dc.w   G_ITALIC|G_SHADOW
455:            .dc.l   mes6
456:            .dc.w   -1
457: *
458: mes0:      .dc.b   '標準(plain)',0
459: mes1:      .dc.b   '強調(bold)',0
460: mes2:      .dc.b   '斜体(italic)',0
461: mes3:      .dc.b   '下線(underline)',0
462: mes4:      .dc.b   '袋文字(outline)',0
463: mes5:      .dc.b   '影つき(shadow)',0
464: mes6:      .dc.b   '組み合わせたりもして',0
465:            .even
466:
467: *
468: *          初期化
469: *
470: reglist reg        d1-d7/a1-a5
471: *
472: init:
473:            movem.l reglist,-(sp)                  *とりあえずレジスタを保存する
474:
475:                                          *コマンド行引数を得る
476:            clr.w   -(sp)                          *argvは作らない
477:            clr.l   -(sp)                          *引数文字列も要らない
478:            pea.l   winsize(WORK)                  *ここにウィンドウの位置を得る
479:            move.l  a2,-(sp)                       *a2=コマンド行
480:            SXCALL  __TSTakeParam                  *
481:            lea.l   14(sp),sp                      *
482:
483:                                          *ウィンドウの表示位置を求める
484:            btst.l  #0,d0                          *'-w～'の指定はあったか?
485:            bne     init0                          *  指定があった
486:                                          *指定がなかったときは
487:            SXCALL  __TSGetWindowPos               *表示位置を適当に決めてもらう
488:            move.l  d0,winsize(WORK)               *左上隅座標をセット
489:            addi.l  #WINSIZE,d0                    *それにウィンドウの大きさを足して
490:            move.l  d0,winsize+4(WORK)             *右下隅座標をセット
491:
492: init0:     SXCALL  __TSGetID                      *自分のタスクIDを得る
493:
494:                                          *ウィンドウをオープンする
495:            move.l  d0,-(sp)                       *タスクID
496:            move.w  #TRUE,-(sp)                    *クローズボックスあり
497:            pea.l   -1.w                           *ウィンドウは一番手前に表示する
498:            move.w  #WINDEFID,-(sp)                *ウィンドウ定義ID
499:            move.w  #TRUE,-(sp)                    *ウィンドウは可視
500:            pea.l   wintitle(pc)                   *ウィンドウのタイトル
501:            pea.l   winsize(WORK)                  *ウィンドウの位置・大きさ
502:            clr.l   -(sp)                          *ウィンドウ構造体を格納する
503:                                                   *  メモリはメモリマネージャまかせ
504:            SXCALL  __WMOpen                       *オープン
505:            lea.l   26(sp),sp                      *
506:            bmi     iniret                         *エラー時はN=1でリターン
507:
508:            move.l  a0,winptr(WORK)                *ウィンドウ構造体へのポインタを
509:                                                   *  しまっておく
510:            move.w  #WINOPT,wOption(a0)            *ウィンドウオプションを設定する
511:
512:            move.l  a0,-(sp)                       *
513:            SXCALL  __GMSetGraph                   *自分の描画環境をカレントグラフにして
514:            SXCALL  __WMDrawGBox                   *グローボックスを描く
515:            addq.l  #4,sp                          *
516:
517:            moveq.l #0,d0                          *正常終了時はN=0でリターン
518:
519: iniret:    movem.l (sp)+,reglist                  *レジスタを復帰する
520:            rts                                    *初期化完了
521:
522: *
523: wintitle:          LASCII  'untitled'             *ウィンドウのタイトル
524:            .even
525: *
526: tail:
527: *
528:            .end    exec_entry
```

実践ウィンドウプログラミング

ライフゲームSXLIFE

Nakamori Akira 中森 章

SXシステムの概要がなんとなくわかったところで，ウィンドウ上で動くアプリケーションを実際に作ってみましょう。第1弾はライフゲームです。プログラムは付録ディスクにも収録されています。暁子さんと並べるなりしてご覧ください。

暁子がすごい。それがSX-WINDOWを初めて見たときの感想でした。これは犬と自転車に乗った暁子さんがウィンドウ内を動き回るというだけの環境ソフトですが，それだけでSX-WINDOWの価値を示すには十分なものでした。また，最近ネットに流れている環境(？)ソフトにマウスカーソルを追いかけてキョロキョロ動く目玉があります。これも見ていて楽しいアプリケーションのひとつです。

SX-WINDOWのようなウィンドウ環境の楽しみ方にはこのような環境ソフトをいくつも画面に並べてその動きを眺めるということがあります。しかし他人が作ったソフトを眺めているだけでは少し物足りません。自分だけの環境ソフトが欲しい。ということで，今回はSX-WINDOW上で動く環境ソフトをひとつ作ってみることにしました。ターゲットはライフゲームです。

プログラムの構造

と，元気よくプログラム作成に取り掛かったのはいいのですが，私にはウィンドウ上でのプログラムを作った経験はありません。まったくゼロの状態からの出発です。そこで，私が最初に行ったのはSX-WINDOW上で動くプログラムの構成を知るということです。まず，目的のライフゲームに構造が最も近いと思われる暁子.XをDIS.Xで逆アセンブルすることから始めました。このほかにも時計.Xやノート.XなどSX-WINDOWについてきたソフトもかたっぱしから逆アセンブルして構造を確かめてみました。

その結果すべてのSX-WINDOW上のプログラムは図1のような構造をしていることが判明したのです。つまり，プログラムはCOMMAND.X（やEXECシステムコール）から起動された場合とSX-WINDOWから起動された場合とで異なる入り口を持っており，それぞれ必要な前処理を行います。そして，初期化（ウィンドウのオープンなど）を行ったあとはイベント待ちの無限ループに陥り，発生したイベントに応じた処理を延々と繰り返すというものです（これはMacintoshのプログラムと同じ構造であることがあとでわかったのですが）。

結局これらの処理はそのまま流用すればよく，私たちは残った部分，すなわちプログラムの初期化処理および各イベントの処理を書けばいいでしょう。結構単純だと思いませんか（少し勇気がわいてきた）。

なお，プログラム起動時の各レジスタの値は次のように設定されているようです。

a1 ： データ領域の先頭アドレス
a2 ： コマンドラインのアドレス
a3 ： 環境のアドレス
a7(usp)： データ領域の最後＋1
d0 ： 自分のタスクID
d1 ： 複製元のタスクID

ここで，データ領域の先頭が，レジスタ(a1)で与えられるところが重要です（データ領域の大きさはプログラムの先頭のヘッダ部分で与える）。すべてのデータ参照をa1レジスタ相対で行って分岐をPC相対で行えば完全にリロケータブルなプログラムを作ることができます（だからどうだと言われてもどういう利点があるのかよくわからないが，きっといいことがあるに違いない）。今回参考にしたプログラムはどれも（C言語で書かれたもの以外は）a1レジスタをa5レジスタにコピーして，a5レジスタ相対でデータを参照していました。これから作るプログラムもこの作法にしたがっておきましょう。

ウィンドウの表示

プログラムの構造が思いのほか簡単で安心したところで次のステップです。ここでは初期化処理を考えます。これはウィンドウをオープンして画面に表示する処理です。これはウィンドウをオープンするためのSX-WINDOWのシステムコールである，

__WMOpen

を使用すればよいでしょう。付録ディスクのドキュメントによると，__WMOpenというシステムコールは8個の引数を持ちます。各引数の意味は，

第1引数：ウィンドウレコードの存在アドレス
第2引数：ウィンドウの位置と大きさ

図1　プログラムの構造

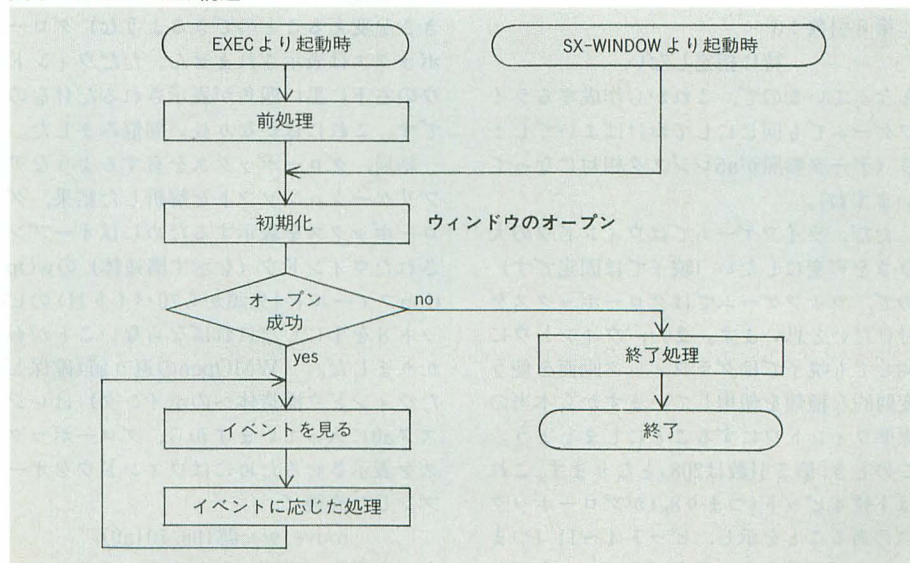
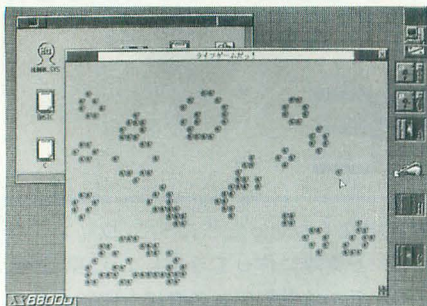



写真1　SXLIFE. X

第3引数：ウィンドウ名へのポインタ
第4引数：ウィンドウ表示の有無
第5引数：ウィンドウの種類
第6引数：生成するウィンドウの
　　　　前後関係
第7引数：クローズボックスの有無
第8引数：ユーザー定義のデータ

となっています（これもMacintoshとほとんど同じだ）。この引数の値は暁子のソースプログラム（DIS. Xによる）では，

第1引数：disp1(a5)
　　　　データ領域のどこか
第2引数：disp2(a5)
　　　　データ領域のどこか
第3引数：disp3(PC)
　　　　プログラム中のどこか
第4引数：$FFFF
　　　　可視
第5引数：$320
　　　　標準ウィンドウ
　　　　（グラフィックサポート，
　　　　　クローズボックスのみ）
第6引数：$FFFFFFFF
　　　　一番手前に表示する
第7引数：$FFFF
　　　　クローズボックスあり
第8引数：0
　　　　特に指定しない

となっているので，これから作成するライフゲームでも同じにしておけばよいでしょう（データ参照がa5レジスタ相対になっていますね）。

ただ，ライフゲームではウィンドウの大きさを可変にしたい（暁子では固定です）ので，ライフゲームではグローボックスを付けたいと思います。また，ウィンドウに関しても暁子ではグラフィック画面を使う変則的な種類を使用していますから本当の標準ウィンドウにすることにしましょう。このとき，第5引数は208_Hとなります。これは下位4ビット（つまり8_H）がグローボックスのあることを示し，ビット4～11（つまり20_H）が標準ウィンドウであることを示しています。

また，ここで注意すべきことは第1引数で示すウィンドウへのポインタが指し示す先は“ウィンドウ”というデータ構造が必要とするだけのバイト数（114バイト）を確保していなければならないということです。もし，第1引数として0が指定された場合はウィンドウのデータ構造のために必要なメモリが勝手に（メモリマネージャによって）確保されるようです。この第1引数に0を与えるかそれ以外のアドレスを与えるかということはウィンドウを消去するときに重要になってきますからしっかりと覚えておきましょう。ここで0を与えた場合はウィンドウの消去に，

__WMDispose

を使用し，それ以外の場合はウィンドウの消去に，

__WMClose

を使用するようです。

最初，私は__WMOpenの第1引数に0を指定しているにも関わらず，無条件に__WMCloseを用いてウィンドウを消去していたため，終了時に決まってアドレスエラーが発生してシステムがデッドロックしていました。すべてに関して参考にしたプログラムの真似をすればよいのに奇をてらって第1引数に0を指定した（余分なメモリ領域を用意したくなかった）ことによる失敗です。定型的な処理に変な独創性を持たせてはいけないという教訓です。

ところで，このような定型的な操作に従ってウィンドウをオープンしても思ったようにことが運ぶとは限りません。問題はグローボックスです。__WMOpen関数の第5引数でグローボックスを指定してもオープンしたウィンドウには（ウィンドウの大きさを変えることのできるような）グローボックスは表示されません。ただウィンドウの右下に黒い四角が表示されるだけなのです。これにはかなり長い間悩みました。

結局，グローボックスを有するようなアプリケーションソフトを解析した結果，グローボックスを表示するためにはオープンされたウィンドウ（を示す構造体）のwOptionフィールド（先頭から70バイト目）のビット8を1にしなければならないことがわかりました。__WMOpenの返り値（確保したウィンドウ構造体へのポインタ）はレジスタa0に入っていますから，グローボックスを表示させるためにはウィンドウをオープンした直後に，

```
move.w  #$108,70(a0)
```

という操作が必要になります。このときソースオペランドで指定するイミディエト値の下4ビットは__WMOpenの第5引数の下4ビットと同じものです。

さて，ウィンドウのオープンについては以上で終わりですが，多くのソフトではこのあとに，

__GMAPage（アクセスするテキスト画面）
__GMForeColor（文字の色）
__GMBackColor（背景の色）
__GMPenMode(文字を書くときの設定)
__GMFontKind(文字フォントの種類)
__GMFontSize(文字フォントの大きさ)

というシステムコールを順番に実行しているようです。この設定はウィンドウをオープンするときの定型処理として覚えておくとよいでしょう。

ところで，__WMOpenではウィンドウをオープンする位置と大きさを指定しなければなりません（第2引数）。この位置は固定値でもよいのですが通常は，

__TSGetWindowPos

というシステムコールで次にウィンドウをオープンするのに都合のよいランダムな位置を得るようにします。さらにSX-WINDOWを一度終了して再び立ち上げる場合，前にあったウィンドウの位置を覚えておいてほしい場合があります。このときは，

__TSTakeParam

というシステムコールを使えば元の位置を取り出せます。このシステムコールは本来はコマンドラインから立ち上げる場合にコマンドへの引数を取り出すためのものですが，こういう使い道もあるのです。

これまでの説明をまとめると，ウィンドウのオープンを行うための処理は図2のようになります。

最低限の処理

ウィンドウがオープンできるようになったからといって気を抜いてはいけません。プログラムの難しさはこれからです。図1のプログラムの構造でも示したように，次はイベント処理です。この処理をしないとウィンドウをオープンしたあと，システムはウンともスンともいわなくなってしまいます（つまりデッドロックね）。

イベントとはシステムで発生するすべてのできごとのことで，マウスのボタンやキーボードを押したりする動作，あるいはウィンドウの上下関係の変更などがあります。これらのイベントが次々とプログラムに通知されてきますから，各プログラムではそ

れに応じた処理をしてやらなければなりません。SX-WINDOWで発生するイベントは次の13種類がすべてです。

アイドル
マウス左ボタンダウン
マウス左ボタンアップ
マウス右ボタンダウン
マウス右ボタンアップ
キーダウン
キーアップ
アップデート
アクティベート
ユーザー定義のイベント1
ユーザー定義のイベント2
ユーザー定義のイベント3
ユーザー定義のイベント4

これだけのイベントが発生するわけですが，これらすべてのイベントに対して処理をする必要はありません。プログラムで必要な処理だけを行えばよいのです。

ただ，どうしても必要な処理というものもあります。それはアップデートとアクティベートです。これらのイベントはウィンドウの状態が変化したことをプログラムに知らせるためのものです。ウィンドウの上下関係が変わったときや前のウィンドウに隠れていた部分が現れたときなどにこれらのイベントが発生します。要するにウィンドウ画面の書き直し要求がこれらのイベントなのです。

ウィンドウの書き直しが行われないと悲惨な状況に陥ることは目に見えていますね。これらのイベントに対する処理はそんなに難しくありません。どのプログラムでも同じようなことをやっているだけですから，私たちがプログラムを作るときもそのまま真似をすればよいでしょう（プログラムの具体例は後ほど示します)。

アクティベート，アップデートそれぞれに対して行うべき処理の概要を図3と図4に示します。図3のアクティベート処理に関してはアプリケーションに関わってくる処理は何ひとつありませんから，どんなプログラムでも同じことをしておけばよいと思います。図4のアップデートに関しても実際に画面を書き直す以外の処理はまったくそのままでかまわないでしょう。

さて，ここまでの処理に関して2つほど注意すべきことがあります。まず，自分のウィンドウに対して何か処理をする場合には必ず，

__GMSetGraph

というシステムコールを実行しなければなりません。これは，早い話が，これからウ

図2　ウィンドウのオープン

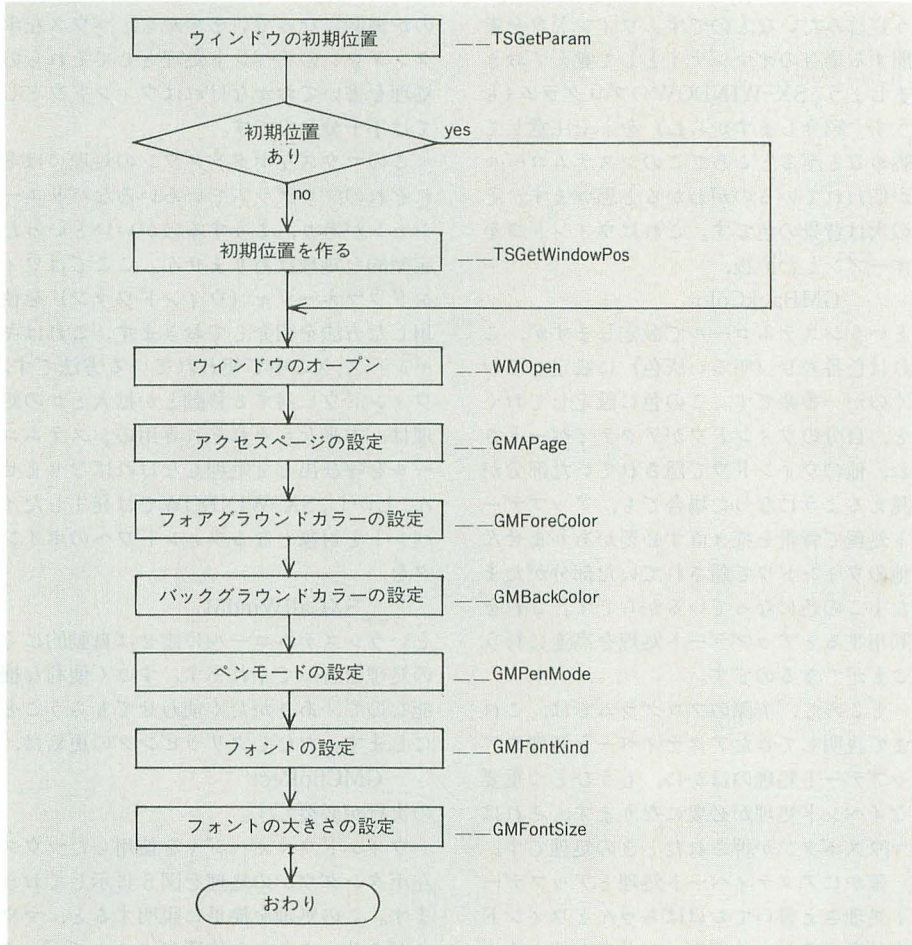
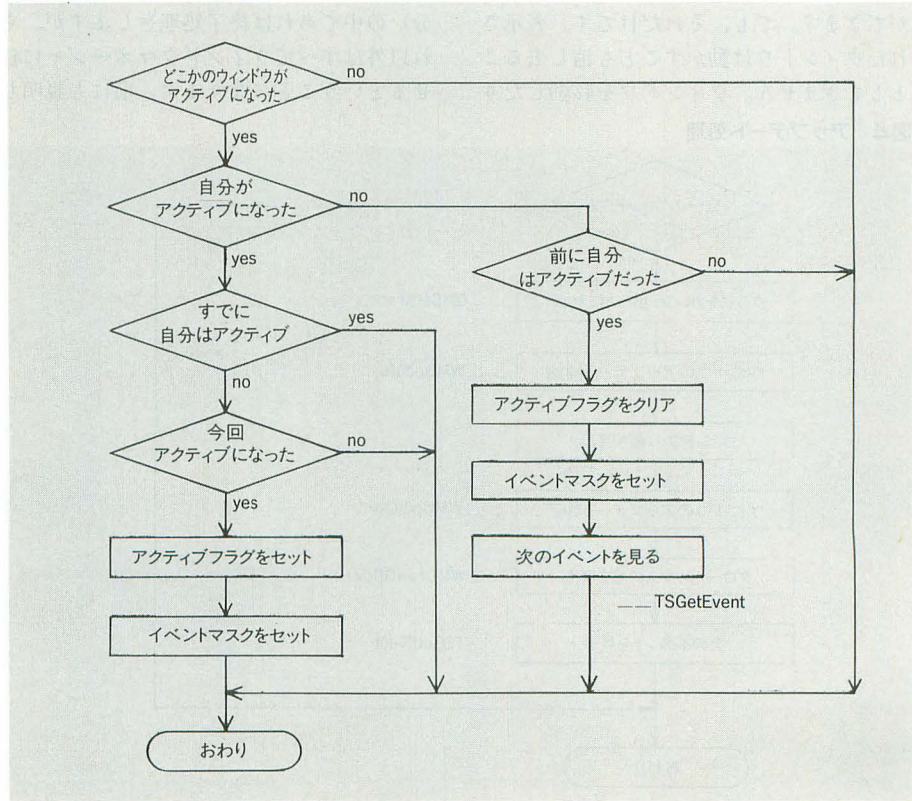



図3　アクティベート処理

ィンドウの内容（画面）を変更するよという宣言みたいなものです。ウィンドウを参照する場合のオマジナイとして覚えておきましょう。SX-WINDOWのプログラム（もうすぐ紹介しますからね）をよく注意して眺めると至るところでこのシステムコールが使われているのがわかると思います。その次は背景の色です。これはウィンドウをオープンした直後，

　__GMBackColor

というシステムコールで設定しますが，これは色番号9（明るい灰色）に設定しておくのが一番楽です。この色に設定しておくと，自分のウィンドウがアクティベートされ，他のウィンドウで隠されていた部分が見えるようになった場合でも，アップデート処理で背景を描き直す必要がありません。他のウィンドウで隠されていた部分がたまたまこの色になっているからです。これを利用するとアップデート処理を高速に行うことができるのです。

ところで，実際のプログラムでは，これまで説明してきたアクティベート処理やアップデート処理のほかに，もうひとつ重要なイベント処理が必要になります。それはマウスボタンが押されたときの処理です。

確かにアクティベート処理とアップデート処理さえ書いておけばちゃんとウィンドウは表示され，他のウィンドウと仲よくマルチタスクしながら画面上に存在することができます。でも，それだけです。表示されたウィンドウは動かすことも消し去ることもできません。ウィンドウを移動したり消し去るためにはマウスの左ボタンを使うのが慣例になっていますから，マウス左ボタンダウンのイベント処理としてそれらの処理を書いておかなければウィンドウとしては不十分なのです。

このマウス左ボタンダウンの処理にはそれぞれのプログラムでいろいろなバリエーションがあり，こうするのがいいといった定型的な処理はありません。ここではウィンドウマネージャ（ウィンドウマン）を使用した方法を紹介しておきます。これはキャンバス.Xの中で使われている方法です。ウィンドウに対する移動とか拡大とかの処理は，本来ならそれぞれ専用のシステムコールを呼び出して処理しなければなりません。しかし，SX-WINDOWでは発生したイベントと対象となるウィンドウへのポインタを，

　__SXCallWindM

というシステムコールに渡せば自動的にその処理を行ってくれます。すごく便利な機能なので，ありがたく使わせてもらうことにします。ただしクリッピングの更新は，

　__GMClipRect

の実行が必要です。

ウィンドウマネージャを使用したマウス左ボタンダウンの処理を図5に示しておきます。この処理を簡単に説明すると，マウスがクリックされた位置がクローズボックス（ウィンドウの右上についている×の部分）の中であれば終了処理をしますが，それ以外はすべてウィンドウマネージャに任せるということになります。前にも説明したとおり，終了処理でのシステムコールは__WMCloseと__WMDisposeの区別をしっかりとしなければなりません。

さて，これまでのまとめとして，ここで簡単なプログラムを紹介しておきましょう。リスト1がそのプログラムですが，これはSX-WINDOWの画面上にのっぺらぼうのウィンドウを表示するプログラムです。一応，アクティベート，アップデート，マウス左ボタンダウンといった最低限のイベント処理は書いてありますから，ウィンドウの移動，拡大，消去は自由自在です。ここまでくれば目的（ライフゲームですよ，忘れないでね）は8割方達成したも同然です。

図4　アップデート処理

ライフゲーム本体

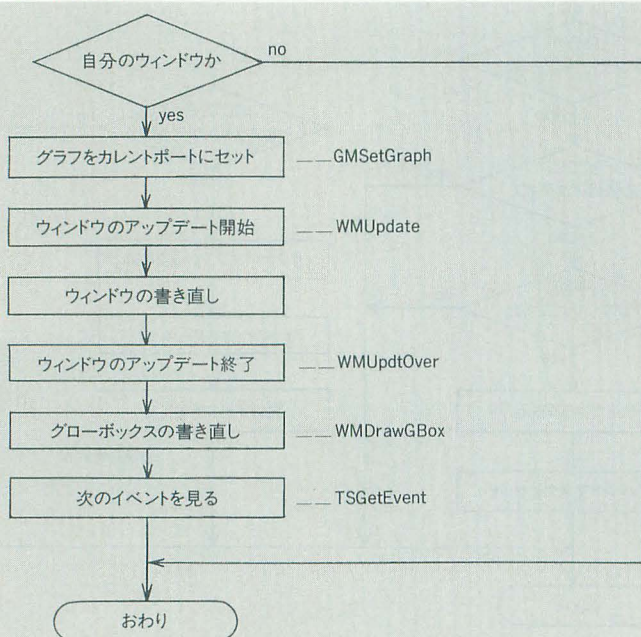
ウィンドウ表示のほうはなんとかメドがつきましたから，本体であるライフゲームについて考えましょう（ライフゲーム自体については参考文献を見てね）。SX-WINDOW上でのライフゲームの基本的な動作は，ある初期状態から始めて，1単位時間経過するごとの状態を次々にウィンドウに表示することになります。

このウィンドウ画面の変更はシステムになにも他のイベントが発生していないとき，すなわちアイドルイベントが発生したとき行います（アップデートイベント発生時にはもちろん画面を書き直す必要がある）。

したがって，このライフゲームでは時分割でウィンドウに与えられてくるアイドルイベントのわずかな時間内でウィンドウ画面の書き換えを行わなければならないため，かなりの高速性が要求されます。とはいえ，1画面の状態が変更されるのを待っていたのでは（10MHzの68000では）どうしても処理が遅くなってしまいます（このとき他のウィンドウの動きも悪くなってしまう）。

そこで考えたのが1回のアイドルイベントにつきウィンドウ画面の1行のみを変更するということです。これによって他のウィンドウの動きに影響を与えることなく，そこそこに高速なウィンドウ画面の書き換えが実現できます（できる予定です）。

それでは今回のライフゲームの1行書き換えアルゴリズムを実現するC言語のプログラムをリスト2（1行の書き換え），リスト3（アップデート処理時の全画面書き換え），リスト4（ライフゲームの初期化処理）に示しておきましょう。リスト2において，fieldという2次元配列が存在/非存在の状態を保持するセルの集まりで，life1という関数が1回の呼び出しごとに引数で与えら

注意）　本紙に掲載したリストにはウィンドウズーム時のクリッピング処理が抜けています（だって気づかなかったんだもん）。でも，付録ディスクのリストでは直してあるのでご安心を。

れたfieldの特定の1行が次の時刻にどのように変化するかを調べています。そして変化のあったセルだけをウィンドウ画面上で書き直すようにしてさらに速さを稼ぐようにしています（実際に書き直すのは引数で与えられたよりもひとつ前の行ですが，なぜそうしなければいけないか考えてみましょう）。

それと，いうまでもなく，psetがウィンドウ画面上に点を打つための関数，presetはウィンドウ画面上の点を消すための関数です。リスト3，リスト4については説明は不要でしょう。というわけで，リスト2，リスト3，リスト4を先に作ったウィンドウ表示のプログラムに組み込めばライフゲームの完成です。

実際の組み込み作業はリスト2，リスト3，リスト4のプログラムをGCCでコンパイルし，出てきたアセンブリ言語ソースを手で変更することで行いました。すべてのデータ参照をa5レジスタ相対に変えてしまったので，コンパイルされた直後のソースリストの面影はあまり残っていません。ただ，リスト3についてはあまりに遅そうだったので最高速で画面の書き直しができるようにアセンブリ言語で1から書き直してしまいました。

このときも背景色が明るい灰色であることを利用して存在点のみを書き直すだけにしています。それ以外に関してはアルゴリズムが複雑であることもあり，特にアセンブリ言語で書き直すことはやっていません。

ところで，pset関数やpreset関数には，

__GMPutRImg

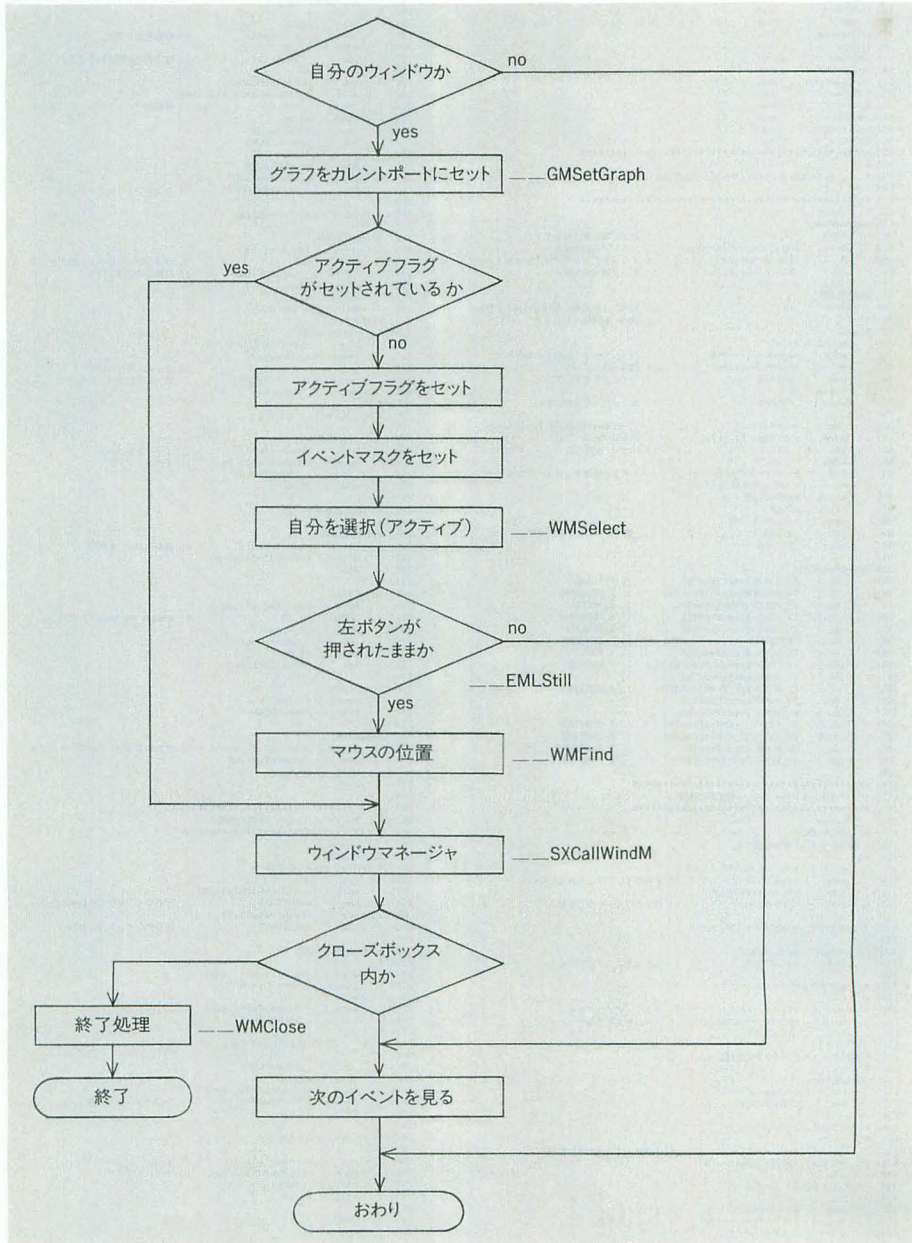
というシステムコールを使用しています。最初は‘*’とか‘●’といった文字をウィンドウに表示することで存在点を表そうと思ったのですが，いろいろと試行するうちに一度ウィンドウに書き込んだ文字を消去するのはかなり面倒だということがわかりました。それで結局パターンをそのまま書き込むことにしたのです。これによって表示するパターンをいろいろとデザインすることができるようになり，このほうがかえってよかったかなと思っています。

パターンの設定機能

ここで完成したライフゲームの全リストをリスト5に示します。このリストでは今までに述べた機能のほかにパターン（ゲーム）の設定機能を持たせてあります。パターンの設定はマウスの右ボタンで行います。ライフゲームのウィンドウ内でマウスの右ボタンが押されるとウィンドウ画面の更新はロックされ（userWRKという変数をフラグに使っている）パターンの設定モードに移ります。あとは，

__EMMSLoc

というシステムコールを使ってマウスの座標を知り，その位置に点を描くだけ（もちろんfieldにも登録する）です。一方，そこに点がある（fieldの内容を参照する）ときにはその点を消去することにします。そして，設定の終了後はマウスの左ボタンのクリックでロックがはずれライフゲームの実行が再開されます。このライフゲームの実行画面を写真1に示しておきましょう。われながら結構いいできになったと思っていますが，みなさんはどう思いますか。

*

なにもかも手探りの状態で作り始めたSX-WINDOW上のライフゲームですが，既存のプログラムの手法を真似ることにより一応の完成を見ました。一見不可能に思えることもやればなんとかなるものなのですねえ。私は毎日のようにライフゲームを眺めてはニタニタしています。みなさんもこのライフゲームをSX-WINDOWの環境ソフトとして仲間に入れてやってください ね。

≪参考文献≫

1) BNN第2企画部（編），インサイドマック徹底ガイド（上巻），BNN，1987年.
2) 有澤誠，レクリエーショナルプログラミングー遊びの中の情報処理，ソフトバンク，1990年.

図5　マウスダウン(左ボタン)処理

リスト1

```
  1:         .include        sx/sxcall.equ
  2:         .include        doscall.mac
  3: DispHight       equ     40
  4: * イベント構造（18バイト）
  5: what    equ     0               ;; イベントの種類
  6: with    equ     2               ;; イベントに関連した引数
  7: when    equ     6               ;; イベントの発生時（システム内部カウント）
  8: where   equ     10              ;; マウスの座標（グローバル座標系）
  9: addtion equ     14              ;; 特殊キーの状態
 10: * データエリア構造
 11: commdLine       equ     0       ;; コマンドライン
 12: envrnment       equ     4       ;; 環境
 13: evntMsk         equ     8       ;; イベントマスク
 14: evntRec         equ     12      ;; イベント (18 Byte)
 15: Window          equ     30      ;; ウインドウ構造 (114 Byte)
 16: bounds          equ     230
 17: leftTop         equ     230
 18: rightBottom     equ     234
 19: title           equ     238
 20: visible         equ     242
 21: wDefID          equ     246
 22: behind          equ     250
 23: cBox            equ     254
 24: taskID          equ     258
 25: wPointer        equ     262
 26: active          equ     266
 27: userWRK         equ     270
 28: * 画面の色
 29: WHITE   equ     %1000
 30: BLACK   equ     %1011
 31: LIGHT   equ     %1001
 32: DARK    equ     %1010
 33: YELLOW  equ     %1100
 34: RED     equ     %1101
 35: GREEN   equ     %1110
 36: BLUE    equ     %1111
 37: * ウインドウパートコード
 38: inDesk          equ     0
 39: inContent       equ     3
 40: inDrag          equ     4
 41: inGrowHi        equ     5
 42: inGrowLo        equ     6
 43: inGoAway        equ     7
 44: inZoomOut       equ     8
 45: inZoomIn        equ     9
 46: inOpt           equ     10
 47: inOneDrag       equ     5
 48: inGrow          equ     6
 49: inClose         equ     7
 50: inWPageUp       equ     10
 51: inWPageDown     equ     11
 52: inClose2        equ     12
 53: inParent        equ     13
 54: inClipOn        equ     14
 55: inClipOff       equ     15
 56: inDrive         equ     16
 57: ************************************************
 58: *
 59: *       SX-Window サンプルプログラム
 60: *
 61: ************************************************
 62:     .text
 63: moduleHead:
 64:     dc.b    'OBJR'                  ;; 再入可能なオブジェクト
 65:     dc.l    moduleEnd-moduleHead    ;; コード領域の大きさ
 66:     dc.l    SXmain-moduleHead       ;; スタートアドレス・オフセット
 67:     dc.l    dataEnd-dataHead        ;; データ領域の大きさ
 68:     dc.l    0,0,0,0
 69: execStart:
 70:     dc.w    _EXIT                   ;; EXEC より起動した場合は何もしない
 71: *                                   ;; Fock 起動時エントリ
 72: SXmain:
 73:     movea.l a1,a5
 74:     move.l  a2,commdLine(a5)        ;; コマンドラインへのポインタ
 75:     move.l  a3,envrnment(a5)        ;; 環境へのポインタ
 76:     bsr     _SXinit                 ;; ウインドウをオープン
 77:     tst.w   d0
 78:     bmi     _SXterm                 ;; エラーだったらおしまい
 79: _SXloop:
 80:     pea.l   evntRec(a5)             ;; Event Record (pointer)
 81:     move.w  evntMsk(a5),-(sp)       ;; Event Mask
 82:     .dc.w   __TSEventAvail          ;; Event を調べる
 83:     addq.l  #6,sp
 84:     lea     EventProcTbl(pc),a1     ;; イベント処理テーブルへのポインタ
 85:     move.w  evntRec+what(a5),d0
 86:     and.w   #$000f,d0
 87:     lsl.w   #2,d0
 88:     move.l  (a1,d0.w),d0
 89:     jsr     (a1,d0.l)               ;; 処理ルーチンを呼び出す
 90:     bra     _SXloop
 91:
 92: EventProcTbl:
 93:     .dc.l   EV_IDLE-EventProcTbl        ;; E_IDLE
 94:     .dc.l   EV_MSLDOWN-EventProcTbl     ;; E_MSLDOWN
 95:     .dc.l   EV_MSLUP-EventProcTbl       ;; E_MSLUP
 96:     .dc.l   EV_NONE-EventProcTbl        ;; E_MSRDOWN
 97:     .dc.l   EV_NONE-EventProcTbl        ;; E_MSRUP
 98:     .dc.l   EV_KEYDOWN-EventProcTbl     ;; E_KEYDOWN
 99:     .dc.l   EV_NONE-EventProcTbl        ;; E_KEYUP
100:     .dc.l   EV_UPDATE-EventProcTbl      ;; E_UPDATE
101:     .dc.l   EV_NONE-EventProcTbl        ;;
102:     .dc.l   EV_ACTIVATE-EventProcTbl    ;; E_ACTIVATE
103:     .dc.l   EV_NONE-EventProcTbl        ;;
104:     .dc.l   EV_NONE-EventProcTbl        ;;
105:     .dc.l   EV_SYSTEM1-EventProcTbl     ;; E_SYSTEM1
106:     .dc.l   EV_SYSTEM2-EventProcTbl     ;; E_SYSTEM2
107:     .dc.l   EV_NONE-EventProcTbl        ;; E_SYSTEM3
108:     .dc.l   EV_NONE-EventProcTbl        ;; E_SYSTEM4
109:
110: ****************************************
111: *   アクティベイト・イベント発生時の処理
112: ****************************************
113:     .even
114: EV_ACTIVATE:
115: * 自分のウインドウかを調べる
116: *
117:     move.l  evntRec+with(a5),d0
118:     beq     retACTIVE               ;; どのウインドウにも該当しない
119:     cmp.l   wPointer(a5),d0
120:     bne     otherWindow             ;; 自分のウインドウじゃない
121:
122: * 自分のウインドウがアクティブになった
123: *
124:     tst.w   active(a5)
125:     bne     retACTIVE               ;; もとよりアクティブだった
126:
127: * 今回アクティブになった
128: *
129:     move.w  #1,active(a5)           ;; アクティブフラグをセット
130:     move.w  #$ffff,evntMsk(a5)      ;; イベントマスクをセット
131:     bra     retACTIVE
132:
133: * 他のウインドウがアクティブになった
134: *
135: otherWindow:
136:     tst.w   active(a5)
137:     beq     retACTIVE
138:
139: * 今回アクティブじゃなくなった
140: *
141:     clr.w   active(a5)              ;; アクティブフラグをねせる
142:     move.w  #$ffff,evntMsk(a5)
143:
144: * イベントレコードをのぞく
145: *
146:     pea     evntRec(a5)             ;; イベントレコード
147:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
148:     .dc.w   __TSGetEvent
149:     addq.l  #6,sp                   ;; イベントマスクをセット
150:
151: retACTIVE:
152:     rts
153: **********************************
154: *   アイドル・イベント発生時の処理
155: **********************************
156:     .even
157: EV_IDLE:
158:     rts
159:
160: ************************************
161: *   キーダウン・イベント発生時の処理
162: ************************************
163:     .even
164: EV_KEYDOWN:
165:     rts                             ;; 何もしない
166:
167: ********************************************
168: *   マウスダウン・イベント時の処理（左ボタン）
169: ********************************************
170:     .even
171: EV_MSLDOWN:
172:     movem.l d1-d7/a1-a5,-(sp)       ;; レジスタを保存
173: *
174: * 自分のウインドウかを調べる
175: *
176:     move.l  evntRec+with(a5),d0
177:     beq     retMSLDOWN              ;; どのウインドウにも該当しない
178:     cmp.l   wPointer(a5),d0
179:     bne     retMSLDOWN              ;; 他のウインドウだった
180: *
181: * カレントポートにセット
182: *
183:     move.l  d0,-(sp)
184:     .dc.w   __GMSetGraph
185:     addq.l  #4,sp
186:     move.l  a0,a2
187:     tst.w   active(a5)
188:     bne     procMSLDOWN             ;; 前からアクティブだった
189:     move.w  #1,active(a5)
190:     move.w  #$ffff,evntMsk(a5)      ;; イベントマスクをセット
191: *
192: * ウインドウを切り替える
193: *
194:     pea     (a2)
195:     .dc.w   __WMSelect              ;; 自分をセレクト
196:     addq.l  #4,sp
197:     .dc.w   __EMLStill              ;; 左ボタンが押されたままか
198:     tst.l   d0
199:     beq     noStillMSLDOWN
200:     move.l  evntRec+where(a5),-(sp)
201:     .dc.w   __WMFind                ;; 座標がウインドウのどこにあるか
202:     addq.l  #4,sp
203:     cmp.w   #inDrag,d0
204:     beq     procMSLDOWN
205:     cmp.w   #inGrow,d0
206:     beq     procMSLDOWN
207:     cmp.w   #inContent,d0
208:     bne     noStillMSLDOWN          ;; コンテントリージョン以外
209: *
210: * ウインドウマネージャーを呼ぶ
211: *
212: procMSLDOWN:
213:     pea     evntRec(a5)
214:     pea     (a2)                    ;; ウインドーレコードへのポインタ
215:     .dc.w   __SXCallWindM           ;; 移動等の処理を行う
216:     addq.l  #8,sp
217:     tst.l   d0
218:     bmi     errMSLDOWN              ;; エラー
219: * サイズ固定
220:     tst.w   d0
221:     beq     noStillMSLDOWN
222:     cmp.w   #inGrow,d0
223:     beq     procClip                ;; よくわからないけど
224:     cmp.w   #inGoAway,d0            ;; 他人のプログラムの真似
225:     beq     _SXgoAway
226: *
227: * イベントレコードをのぞく
228: *
229: noStillMSLDOWN:
230:     pea     evntRec(a5)             ;; イベントレコード
231:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
232:     .dc.w   __TSGetEvent
233:     addq.l  #6,sp
234: *
235: * 今回の状態をセーブ
236: *
237: retMSLDOWN:
238:     clr.b   userWRK(a5)             ;; 描画のロックを解除
239:     movem.l (sp)+,d1-d7/a1-a5
240:     rts
241:
242: procClip:
243:     move.l  wPointer(a5),d0
244:     addq.l  #4,d0                   ;; graph rectangle 部分
245:     move.l  d0,-(sp)
246:     .dc.w   __GMClipRect
247:     addq.l  #4,sp
248:     bra     noStillMSLDOWN
249:
250: errMSLDOWN:
251:     moveq.l #-1,d0
252:     bra     retMSLDOWN
253:
254: _SXgoAway:
255:     movem.l (sp)+,d1-d7/a1-a5       ;; クローズボックスが押されたとき
256:     lea     _SXterm(pc),a0
257:     move.l  a0,(sp)
258:     rts
259:
260: **********************************
261: *   アップデート・イベント時の処理
262: **********************************
263:     .even
264: EV_UPDATE:
265: * 自分のウインドウかを調べる
266: *
267:     move.l  evntRec+with(a5),d0
268:     beq     retUPDATE               ;; どのウインドウにも該当しない
269:     cmp.l   wPointer(a5),d0
270:     bne     retUPDATE               ;; 自分のウインドウじゃない
271:
272: * アップデート開始
273: *
274:     move.l  wPointer(a5),-(sp)
275:     .dc.w   __GMSetGraph
276:     addq.l  #4,sp
277:     move.l  wPointer(a5),-(sp)
278:     .dc.w   __WMUpdate
279: *
280: *           ここに書き直し処理を書く
281: *
282:     .dc.w   __WMUpdtOver
283:     addq.l  #4,sp
284:     move.l  wPointer(a5),-(sp)
285:     .dc.w   __WMDrawGBox            ;; グローボックスの書き直し
286:     addq.l  #4,sp
287: *
288: * イベントレコードをのぞく
289: *
290:     pea     evntRec(a5)             ;; イベントレコード
291:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
292:     .dc.w   __TSGetEvent
293:     addq.l  #6,sp
294:
295: retUPDATE:
296:     rts
297:
298: ******************************
299: *   その他のイベント時の処理
300: ******************************
301:     .even
302: EV_MSLUP:
303: EV_NONE:
304: EV_SYSTEM1:
305: EV_SYSTEM2:
306: EV_SYSTEM3:
307: EV_SYSTEM4:
308:     rts                             ;; 何もしません
309:
310: ********************************
311: *   ウインドウのオープンエラー
312: ********************************
313:     .even
314: OpenError:
315:     pea     mess(pc)
316:     move.w  #1,-(sp)
317:     dc.w    __DMError
318:     addq.l  #6,sp
319:     bra     _SXterm
320:
321: ****************************************
322: *   ウインドウのオープンその他の初期化
323: ****************************************
324:     .even
325: _SXinit:
326:     clr.w   -(sp)
327:     clr.l   -(sp)
328:     pea.l   leftTop(a5)             ;; ウインドウ位置
329:     pea.l   (a2)                    ;; コマンドライン
330:     .dc.w   __TSTakeParam           ;; ウインドウの初期位置を取り出す
331:     lea.l   14(sp),sp
332:     btst.l  #0,d0
333:     bne     openWindow              ;; ウインドウの位置指定があった
334:     .dc.w   __TSGetWindowPos        ;; 次にウインドウをオープンする場所
335:     move.l  d0,leftTop(a5)          ;; 左上の座標
336:     add.l   #$01000080,d0           ;; ウインドウの大きさ（横，縦）
337:     move.l  d0,rightBottom(a5)
338: *
339: * ウインドウのオープン
340: *
341: openWindow:
342:     clr.l   -(sp)                   ;; タスクID（意味なし）
343:     move.w  #$ffff,-(sp)            ;; クローズ・ボックスあり
344:     move.l  #$ffffffff,-(sp)        ;; 一番前にウインドウを開く
345:     move.w  #$0208,-(sp)            ;; resource ID<<4+Option 標準ウインドウ
346:     move.w  #$ffff,-(sp)            ;; 可視
347:     pea.l   wtitle(pc)              ;; タイトル
348:     pea.l   bounds(a5)              ;; ウインドウの位置と大きさへのポインタ
349:     pea.l   Window(a5)              ;; ウインドウへのポインタ
350:     .dc.w   __WMOpen
351:     lea.l   26(sp),sp
352:     tst.l   d0
353:     bmi     OpenError
354:     move.l  a0,wPointer(a5)
355:     move.w  #$108,$46(a0)
356:     pea.l   (a0)
357:     .dc.w   __GMSetGraph
358:     addq.l  #4,sp
359:     .dc.w   __TSGetID
360:     move.l  d0,-(sp)
361:     .dc.w   __WMTIDSet
362:     addq.l  #4,sp
363: ************************************************
364: *   描画関連の初期化（関係ないこともやっている）
365: ************************************************
366:     move.w  #%1111,-(sp)
367:     .dc.w   __GMAPage               ;; アクセスビット（プレーン）のセット
368:     addq.l  #2,sp                   ;; current bit map の access page (0,1)
369:     move.w  #BLUE,-(sp)
370:     .dc.w   __GMForeColor           ;; フォアグラウンドカラーのセット
371:     addq.l  #2,sp
372:     move.w  #LIGHT,-(sp)
373:     .dc.w   __GMBackColor           ;; バックグラウンドカラーのセット
374:     addq.l  #2,sp
375:     clr.w   -(sp)                   ;; フォアグラウンドカラー & PSET
376:     .dc.w   __GMPenMode
377:     addq.l  #2,sp
378:     clr.w   -(sp)
379:     .dc.w   __GMFontKind            ;; フォントのセット（初期値：ROM 12-dot)
380:     addq.l  #2,sp
381:     move.l  #12<<16+12,-(sp)        ;; H(12)×V(12)
382:     .dc.w   __GMFontSize            ;; フォントサイズのセット
383:     addq.l  #4,sp
384:     move.w  #$ffff,evntMsk(a5)
385:     moveq.l #$00,d0
386:     rts
387:
388: ************************
389: *   ウインドウの終了処理
390: ************************
391:     .even
392: _SXterm:
393:     move.l  wPointer(a5),d0
394:     beq     __Exit
395:     move.l  d0,-(sp)
396: *       .dc.w   __WMDispose         ;; WMOpenでWindowに0を指定した場合
397:     .dc.w   __WMClose               ;; WMOpenでWindowに0以外を設定した場合
398:     addq.l  #4,sp
399: __Exit:
400:     .dc.w   __TSExit
401:
402:     .even
403: mess:
404:     .dc.b   'エラーでっせ。えらーいこっちゃ！',0
405:
406:     .even
407: wtitle:
408:     .dc.b   20,'標準ウインドウだっ！'
409:
410:     .even
411: moduleEnd:
412:
413:     .offset 0
414: dataHead:
415:     ds.b    4096*2                  ;; スタック用
416: dataEnd:
417:     .text
418:     .end    execStart
419:
```

▶うーむ，クリスタルポルシェ……。なんて高価な消しゴム（違うって）。

中元寺　康之(23)岡山県

リスト2

```
 1: #define XSIZ    64
 2: #define YSIZ    64
 3: char field[XSIZ][YSIZ];
 4: char ys0[YSIZ];
 5: char vs0[YSIZ];
 6: char ys1[YSIZ];
 7: char vs1[YSIZ];
 8: char *vp0;
 9: char *yp0;
10: char *vp1;
11: char *yp1;
12: int  nv0;
13: int  nv1;
14:
15: life1(x)
16: register int x;
17: {
18:         int arnd;
19:         int i,y;
20:         nv1=-1;
21:         for(y=1;y<YSIZ-1;y++){ /* x */
22:                 arnd=field[x][y+1]+field[x][y-1]
23:                     +field[x+1][y]+field[x-1][y]
24:                     +field[x+1][y+1]+field[x-1][y-1]
25:                     +field[x-1][y+1]+field[x+1][y-1];
26:                 if(arnd!=3 && field[x][y]==0 ) continue;
27:                 if(arnd==2 && field[x][y]==1 ) continue;
28:                 if(arnd==3 && field[x][y]==1 ) continue;
29:                 nv1++;
30:                 vp1[nv1]=(field[x][y]==0);
31:                 yp1[nv1]=y;
32:         }
33:         x--;
34:         for(i=0;i<=nv0;i++){
35:                 y=yp0[i];
36:                 if(field[x][y]=vp0[i]) pset(x,y);
37:                 else                    preset(x,y);
38:
39:         }
40:         i=vp0;vp0=vp1;vp1=i;
41:         i=yp0;yp0=yp1;yp1=i;
42:         nv0=nv1;
43: }
```

リスト3

```
 1: #define DispHight 40
 2: char field[64][64];
 3:
 4: life2()
 5: {
 6:    int x,y;
 7:    for(x=0;x<DispHight;x++)
 8:      for(y=0;YSIZ;y++)
 9:          if(field[x][y]) pset(x,y);
10: }
```

リスト4

```
 1: #define DispHight 40
 2: char iniPat[];
 3: char field[64][64];
 4:
 5: initLife()
 6: {
 7:         int i,x,y;
 8:
 9:         for(x=1;x<DispHight;x++)
10:            for(y=1;y<63;y++)
11:                 field[x][y]=0;
12:         i=0;
13:         while(1){
14:                 x=iniPat[i++];
15:                 y=iniPat[i++];
16:                 if(y<0) break;
17:                 field[x][y]=1;
18:                 pset(x,y);
19:         }
20: }
```

リスト5

```
  1:          .include          sx/sxcall.equ
  2:          .include          doscall.mac
  3: DispHight          equ     40
  4: * イベント構造（18バイト）
  5: what     equ     0                 ;; イベントの種類
  6: with     equ     2                 ;; イベントに関連した引数
  7: when     equ     6                 ;; イベントの発生時（システム内部カウント）
  8: where    equ     10                ;; マウスの座標（グローバル座標系）
  9: addtion  equ     14                ;; 特殊キーの状態
 10: * データエリア構造
 11: commdLine          equ     0       ;; コマンドライン
 12: envrnment          equ     4       ;; 環境
 13: evntMsk            equ     8       ;; イベントマスク
 14: evntRec            equ     12      ;; イベント (18 Byte)
 15: Window             equ     30      ;; ウインドウ構造 (114 Byte)
 16: bounds             equ     230
 17: leftTop            equ     230
 18: rightBottom        equ     234
 19: title              equ     238
 20: visible            equ     242
 21: wDefID             equ     246
 22: behind             equ     250
 23: cBox               equ     254
 24: taskID             equ     258
 25: wPointer           equ     262
 26: active             equ     266
 27: userWRK            equ     270
 28: * ライフゲーム用ワーク
 29: _xhold   equ     300
 30: _nv1     equ     _xhold+4
 31: _nv0     equ     _nv1+4
 32: _yp1     equ     _nv0+4
 33: _yp0     equ     _yp1+4
 34: _vp1     equ     _yp0+4
 35: _vp0     equ     _vp1+4
 36: _ys1     equ     _vp0+4
 37: _ys0     equ     _ys1+64
 38: _vs1     equ     _ys0+64
 39: _vs0     equ     _vs1+64
 40: _field   equ     _vs0+64
 41: _next    equ     _field+4096
 42: * 画面の色
 43: WHITE    equ     %1000
 44: BLACK    equ     %1011
 45: LIGHT    equ     %1001
 46: DARK     equ     %1010
 47: YELLOW   equ     %1100
 48: RED      equ     %1101
 49: GREEN    equ     %1110
 50: BLUE     equ     %1111
 51: * ウインドウパートコード
 52: inDesk             equ     0
 53: inContent          equ     3
 54: inDrag             equ     4
 55: inGrowHi           equ     5
 56: inGrowLo           equ     6
 57: inGoAway           equ     7
 58: inZoomOut          equ     8
 59: inZoomIn           equ     9
 60: inOpt              equ     10
 61: inOneDrag          equ     5
 62: inGrow             equ     6
 63: inClose            equ     7
 64: inWPageUp          equ     10
 65: inWPageDown        equ     11
 66: inClose2           equ     12
 67: inParent           equ     13
 68: inClipOn           equ     14
 69: inClipOff          equ     15
 70: inDrive            equ     16
 71: ****************************************************
 72: *
 73: *    SX-Window サンプルプログラム
 74: *
 75: *    ライフゲームだよ
 76: *
 77: *                          1990.11.12, 中森 泉
 78: *
 79: ****************************************************
 80:      .text
 81: moduleHead:
 82:      dc.b     'OBJR'                       ;; 再入可能なオブジェクト
 83:      dc.l     moduleEnd-moduleHead         ;; コード領域の大きさ
 84:      dc.l     SXmain-moduleHead            ;; スタートアドレス・オフセット
 85:      dc.l     dataEnd-dataHead             ;; データ領域の大きさ
 86:      dc.l     0,0,0,0
 87: execStart:
 88:      dc.w     _EXIT                        ;; EXEC より起動した場合は何もしない
 89:      dc.b     'Life Game Written by 中森 泉, Nov. 12, 1990'
 90:      .even
 91: *                                          ;; Fock 起動時エントリ
 92: SXmain:
 93:      movea.l a1,a5
 94:      move.l  a2,commdLine(a5)              ;; コマンドラインへのポインタ
 95:      move.l  a3,envrnment(a5)              ;; 環境へのポインタ
 96:      bsr     _SXinit                       ;; ウインドウをオープン
 97:      tst.w   d0
 98:      bmi     _SXterm                       ;; エラーだったらおしまい
 99: _SXloop:
100:      pea.l   evntRec(a5)                   ;; Event Record (pointer)
101:      move.w  evntMsk(a5),-(sp)             ;; Event Mask
102:      .dc.w   __TSEventAvail                ;; Event を調べる
103:      addq.l  #6,sp
104:      lea     EventProcTbl(pc),a1           ;; イベント処理テーブルへのポインタ
105:      move.w  evntRec+what(a5),d0
106:      and.w   #$000f,d0
107:      lsl.w   #2,d0
108:      move.l  (a1,d0.w),d0
109:      jsr     (a1,d0.l)                     ;;  処理ルーチンを呼び出す
110:      bra     _SXloop
111:
112: EventProcTbl:
113:      .dc.l   EV_IDLE-EventProcTbl          ;; E_IDLE
114:      .dc.l   EV_MSLDOWN-EventProcTbl       ;; E_MSLDOWN
115:      .dc.l   EV_MSLUP-EventProcTbl         ;; E_MSLUP
116:      .dc.l   EV_MSRDOWN-EventProcTbl       ;; E_MSRDOWN
117:      .dc.l   EV_NONE-EventProcTbl          ;; E_MSRUP
118:      .dc.l   EV_KEYDOWN-EventProcTbl       ;; E_KEYDOWN
119:      .dc.l   EV_NONE-EventProcTbl          ;; E_KEYUP
120:      .dc.l   EV_UPDATE-EventProcTbl        ;; E_UPDATE
121:      .dc.l   EV_NONE-EventProcTbl          ;;
122:      .dc.l   EV_ACTIVATE-EventProcTbl      ;; E_ACTIVATE
123:      .dc.l   EV_NONE-EventProcTbl          ;;
124:      .dc.l   EV_NONE-EventProcTbl          ;;
125:      .dc.l   EV_SYSTEM1-EventProcTbl       ;; E_SYSTEM1
126:      .dc.l   EV_SYSTEM2-EventProcTbl       ;; E_SYSTEM2
127:      .dc.l   EV_NONE-EventProcTbl          ;; E_SYSTEM3
128:      .dc.l   EV_NONE-EventProcTbl          ;; E_SYSTEM4
129:
130: ****************************************
131: *    アクティベイト・イベント発生時の処理
132: ****************************************
133:      .even
134: EV_ACTIVATE:
135: * 自分のウインドウかを調べる
136: *
137:      move.l  evntRec+with(a5),d0
138:      beq     retACTIVE                     ;; どのウインドウにも該当しない
139:      cmp.l   wPointer(a5),d0
140:      bne     otherWindow                   ;; 自分のウインドウじゃない
141:
142: * 自分のウインドウがアクティブになった
143: *
144:      tst.w   active(a5)
145:      bne     retACTIVE                     ;; もとよりアクティブだった
146:
147: * 今回アクティブになった
148: *
149:      move.w  #1,active(a5)                 ;; アクティブフラグをセット
150:      move.w  #$ffff,evntMsk(a5)            ;; イベントマスクをセット
151:      bra     retACTIVE
152:
153: * 他のウインドウがアクティブになった
154: *
155: otherWindow:
156:      tst.w   active(a5)
```

▶RPGを作ろうと思った。それで背景を作って動かした。ふと，シューティングが作りたくなった。そして，眠たくなった。結局なにもできなかった。パソコン歴３カ月の私……。

日並　大輔(17)兵庫県

```
157:     beq     retACTIVE
158:
159: * 今回アクティブじゃなくなった
160: *
161:     clr.w   active(a5)              ;; アクティブフラグをねせる
162:     move.w  #$ffff,evntMsk(a5)
163:
164: * イベントレコードをのぞく
165: *
166:     pea     evntRec(a5)             ;; イベントレコード
167:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
168:     .dc.w   __TSGetEvent
169:     addq.l  #6,sp                   ;; イベントマスクをセット
170:
171: retACTIVE:
172:     rts
173: ***********************************
174: *   アイドル・イベント発生時の処理
175: ***********************************
176:     .even
177: EV_IDLE:
178:     tst.b   userWRK(a5)
179:     bne     idlLock
180:     move.l  wPointer(a5),-(sp)
181:     .dc.w   __GMSetGraph
182:     addq.l  #4,sp
183:     move.l  _xhold(a5),d0           ;; X座標
184:     cmp.b   #DispHight,d0
185:     bcs     idlSkip
186:     moveq.l #1,d0
187: idlSkip:
188:     move.l  d0,-(sp)
189:     addq.l  #1,d0
190:     move.l  d0,_xhold(a5)
191:     bsr     lifel                   ;; 1行分の画面を書き直す
192:     move.l  wPointer(a5),-(sp)      ;; グローボックスを書き直す
193:     .dc.w   __WMDrawGBox
194:     addq.l  #8,sp
195: idlLock:
196:     rts
197:
198: ************************************
199: *   キーダウン・イベント発生時の処理
200: ************************************
201:     .even
202: EV_KEYDOWN:
203:     rts                             ;; 何もしない
204:
205: ***********************************************
206: *   マウスダウン・イベント時の処理(左ボタン)
207: ***********************************************
208:     .even
209: EV_MSLDOWN:
210:     movem.l d1-d7/a1-a5,-(sp)       ;; レジスタを保存
211: *
212: * 自分のウインドウかを調べる
213: *
214:     move.l  evntRec+with(a5),d0
215:     beq     retMSLDOWN              ;; どのウインドウにも該当しない
216:     cmp.l   wPointer(a5),d0
217:     bne     retMSLDOWN              ;; 他のウインドウだった
218: *
219: * カレントポートにセット
220: *
221:     move.l  d0,-(sp)
222:     .dc.w   __GMSetGraph
223:     addq.l  #4,sp
224:     move.l  a0,a2
225:     tst.w   active(a5)
226:     bne     procMSLDOWN             ;; 前からアクティブだった
227:     move.w  #1,active(a5)
228:     move.w  #$ffff,evntMsk(a5)      ;; イベントマスクをセット
229: *
230: * ウインドウを切り替える
231: *
232:     pea     (a2)
233:     .dc.w   __WMSelect              ;; 自分をセレクト
234:     addq.l  #4,sp
235:     .dc.w   __EMLStill              ;; 左ボタンが押されたままか
236:     tst.l   d0
237:     beq     noStillMSLDOWN
238:     move.l  evntRec+where(a5),-(sp)
239:     .dc.w   __WMFind                ;; 座標がウインドウのどこにあるか
240:     addq.l  #4,sp
241:     cmp.w   #inDrag,d0
242:     beq     procMSLDOWN
243:     cmp.w   #inGrow,d0
244:     beq     procMSLDOWN
245:     cmp.w   #inContent,d0
246:     bne     noStillMSLDOWN          ;; コンテントリージョン以外
247: *
248: * ウインドウマネージャーを呼ぶ
249: *
250: procMSLDOWN:
251:     pea     evntRec(a5)
252:     pea     (a2)                    ;; ウインドーレコードへのポインタ
253:     .dc.w   __SXCallWindM           ;; 移動等の処理を行う
254:     addq.l  #8,sp
255:     tst.l   d0
256:     bmi     errMSLDOWN              ;; エラー
257: * サイズ固定
258:     tst.w   d0
259:     beq     noStillMSLDOWN
260:     cmp.w   #inGrow,d0
261:     beq     procClip                ;; よくわからないけど
262:     cmp.w   #inGoAway,d0            ;; 他人のプログラムの真似
263:     beq     _SXgoAway
264: *
265: * イベントレコードをのぞく
266: *
267: noStillMSLDOWN:
268:     pea     evntRec(a5)             ;; イベントレコード
269:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
270:     .dc.w   __TSGetEvent
271:     addq.l  #6,sp
272: *
273: * 今回の状態をセーブ
274: *
275: retMSLDOWN:
276:     clr.b   userWRK(a5)             ;; 描画のロックを解除
277:     movem.l (sp)+,d1-d7/a1-a5
278:     rts
279:
280: procClip:
281:     move.l  wPointer(a5),d0
282:     addq.l  #4,d0                   ;; graph rectangle 部分
283:     move.l  d0,-(sp)
284:     .dc.w   __GMClipRect
285:     addq.l  #4,sp
286:     bra     noStillMSLDOWN
287:
288: errMSLDOWN:
289:     moveq.l #-1,d0
290:     bra     retMSLDOWN
291:
292: _SXgoAway:
293:     movem.l (sp)+,d1-d7/a1-a5       ;; クローズボックスが押されたとき
294:     lea     _SXterm(pc),a0
295:     move.l  a0,(sp)
296:     rts
297:
298: ***********************************************
299: *   マウスダウン・イベント時の処理(右ボタン)
300: ***********************************************
301:     .even
302: EV_MSRDOWN:
303:     movem.l d1-d7/a1-a5,-(sp)       ;; レジスタを保存
304: *
305: * 自分のウインドウかを調べる
306: *
307:     move.l  evntRec+with(a5),d0
308:     beq     retMSRDOWN              ;; どのウインドウにも該当しない
309:     cmp.l   wPointer(a5),d0
310:     bne     retMSRDOWN              ;; 他のウインドウだった
311: *
312: * カレントポートにセット
313: *
314:     move.l  d0,-(sp)
315:     .dc.w   __GMSetGraph
316:     addq.l  #4,sp
317:     move.l  a0,a2
318:     tst.w   active(a5)
319:     bne     procMSRDOWN             ;; 前からアクティブだった
320:     move.w  #1,active(a5)
321:     move.w  #$ffff,evntMsk(a5)      ;; イベントマスクをセット
322: *
323: * ウインドウを切り替える
324: *
325:     pea     (a2)
326:     .dc.w   __WMSelect              ;; 自分をセレクト
327:     addq.l  #4,sp
328:     .dc.w   __EMRStill              ;; 右ボタンが押されたままか
329:     tst.l   d0
330:     beq     noStillMSRDOWN
331:     move.l  evntRec+where(a5),-(sp)
332:     .dc.w   __WMFind                ;; 座標がウインドウのどこにあるか
333:     addq.l  #4,sp
334:     cmp.w   #inContent,d0
335:     bne     noStillMSRDOWN          ;; コンテントリージョン以外
336: *
337: * マウスの座標を知る
338: *
339: procMSRDOWN:
340:     move.b  #1,userWRK(a5)          ;; 描画をロックする
341:     .dc.w   __EMMSLoc
342:     tst.l   d0
343:     bmi     errMSRDOWN
344:     move.l  d0,d1
345:     moveq.l #0,d2
346:     move.w  #0,d1
347:     swap    d1                      ;; 水平
348:     move.w  d0,d2                   ;; 垂直
349:     divu    #12,d2
350:     divu    #12,d1
351:     move.w  d1,-(sp)
352:     move.w  d2,-(sp)
353:     lea     _field(a5),a1
354:     lsl.l   #6,d2
355:     lea     (a1,d2.w),a1
356:     lea     (a1,d1.w),a1
357:     move.l  wPointer(a5),-(sp)
358:     .dc.w   __GMSetGraph
359:     addq.l  #4,sp
360:     tst.b   (a1)
361:     beq     nonExist
362:     bsr     preset
363:     clr.b   (a1)
364:     bra     cmnMSRDWN
365: nonExist:
366:     bsr     pset
367:     move.b  #1,(a1)
368: cmnMSRDWN:
369:     addq.l  #4,sp
370:     move.l  wPointer(a5),-(sp)      ;; グローボックスを書き直す
371:     .dc.w   __WMDrawGBox
372:     addq.l  #4,sp
373: *
374: * イベントレコードをのぞく
375: *
376: noStillMSRDOWN:
377:     pea     evntRec(a5)             ;; イベントレコード
378:     move.w  evntMsk(a5),-(sp)       ;; イベントマスク
379:     .dc.w   __TSGetEvent
380:     addq.l  #6,sp
381: *
382: * 今回の状態をセーブ
383: *
384: retMSRDOWN:
385:     movem.l (sp)+,d1-d7/a1-a5
386:     rts
387:
388: errMSRDOWN:
389:     moveq.l #-1,d0
390:     bra     retMSRDOWN
391:
392:
393: **********************************
394: *   アップデート・イベント時の処理
395: **********************************
396:     .even
397: EV_UPDATE:
398: * 自分のウインドウかを調べる
399: *
400:     move.l  evntRec+with(a5),d0
401:     beq     retUPDATE               ;; どのウインドウにも該当しない
402:     cmp.l   wPointer(a5),d0
403:     bne     retUPDATE               ;; 自分のウインドウじゃない
404:
405: * アップデート開始
406: *
407:     move.l  wPointer(a5),-(sp)
408:     .dc.w   __GMSetGraph
409:     addq.l  #4,sp
410:     move.l  wPointer(a5),-(sp)
411:     .dc.w   __WMUpdate
412:     bsr     life2           ;; 画面を書き換える
413:     .dc.w   __WMUpdtOver
414:     addq.l  #4,sp
415:     move.l  wPointer(a5),-(sp)
416:     .dc.w   __WMDrawGBox            ;; グローボックスの書き直し
417:     addq.l  #4,sp
418: *
```

▶X68000グッズのジッポライターのツタンカーメンの顔は目が垂れているので福の神に見える。　影山　秀和(18)広島県

```
419: * イベントレコードをのぞく
420: *
421:     pea     evntRec(a5)          ;; イベントレコード
422:     move.w  evntMsk(a5),-(sp)    ;; イベントマスク
423:     .dc.w   __TSGetEvent
424:     addq.l  #6,sp
425:
426: retUPDATE:
427:     rts
428:
429: ****************************
430: *   その他のイベント時の処理
431: ****************************
432:     .even
433: EV_MSLUP:
434: EV_NONE:
435: EV_SYSTEM1:
436: EV_SYSTEM2:
437: EV_SYSTEM3:
438: EV_SYSTEM4:
439:     rts                          ;; 何もしません
440:
441: *******************************
442: *   ウインドウのオープンエラー
443: *******************************
444:     .even
445: OpenError:
446:     pea     mess(pc)
447:     move.w  #1,-(sp)
448:     dc.w    __DMError
449:     addq.l  #6,sp
450:     bra     _SXterm
451:
452: ****************************************
453: *   ウインドウのオープンその他の初期化
454: ****************************************
455:     .even
456: _SXinit:
457:     clr.w   -(sp)
458:     clr.l   -(sp)
459:     pea.l   leftTop(a5)          ;; ウインドウ位置
460:     pea.l   (a2)                 ;; コマンドライン
461:     .dc.w   __TSTakeParam        ;; ウインドウの初期位置を取り出す
462:     lea.l   14(sp),sp
463:     btst.l  #0,d0
464:     bne     openWindow           ;; ウインドウの位置指定があった
465:     .dc.w   __TSGetWindowPos     ;; 次にウインドウをオープンする場所
466:     move.l  d0,leftTop(a5)       ;; 左上の座標
467:     add.l   #$01000080,d0        ;; ウインドウの大きさ(横，縦)
468:     move.l  d0,rightBottom(a5)
469: *
470: * ウインドウのオープン
471: *
472: openWindow:
473:     clr.l   -(sp)                ;; タスクID(意味なし)
474:     move.w  #$ffff,-(sp)         ;; クローズ・ボックスあり
475:     move.l  #$ffffffff,-(sp)     ;; 一番前にウインドウを開く
476:     move.w  #$0208,-(sp)         ;; resource ID<<4 + Option  標準ウインドウ
477:     move.w  #$ffff,-(sp)         ;; 可視
478:     pea.l   wtitle(pc)           ;; タイトル
479:     pea.l   bounds(a5)           ;; ウインドウの位置と大きさへのポインタ
480:     pea.l   Window(a5)           ;; ウインドウへのポインタ
481:     .dc.w   __WMOpen
482:     lea.l   26(sp),sp
483:     tst.l   d0
484:     bmi     OpenError
485:     move.l  a0,wPointer(a5)
486:     move.w  #$108,$46(a0)
487:     pea.l   (a0)
488:     .dc.w   __GMSetGraph
489:     addq.l  #4,sp
490:     .dc.w   __TSGetID
491:     move.l  d0,-(sp)
492:     .dc.w   __WMTIDSet
493:     addq.l  #4,sp
494: **************************************************
495: *   描画関連の初期化(関係ないこともやっている)
496: **************************************************
497:     bsr     penEnviron
498:     bsr     initLife             ;; ライフゲームの初期化
499:     clr.l   userWRK(a5)          ;; 右ボタン使用時のロック用
500:     clr.l   userWRK+4(a5)        ;; 何かのために(意味なし)
501:     move.w  #$ffff,evntMsk(a5)
502:     moveq.l #$00,d0
503:     rts
504: *************************************************************************
******
505: penEnviron:
506:     move.w  #%1111,-(sp)
507:     .dc.w   __GMAPage            ;; アクセスビット(プレーン)のセット
508:     addq.l  #2,sp                ;; current bit map の access page (0,1)
509:     move.w  #BLUE,-(sp)
510:     .dc.w   __GMForeColor        ;; フォアグラウンドカラーのセット
511:     addq.l  #2,sp
512:     move.w  #LIGHT,-(sp)
513:     .dc.w   __GMBackColor        ;; バックグラウンドカラーのセット
514:     addq.l  #2,sp
515:     clr.w   -(sp)                ;; フォアグラウンドカラー & PSET
516:     .dc.w   __GMPenMode
517:     addq.l  #2,sp
518:     clr.w   -(sp)
519:     .dc.w   __GMFontKind         ;; フォントのセット (初期値:ROM 12-dot)
520:     addq.l  #2,sp
521:     move.l  #12<<16+12,-(sp)     ;; H(12)×V(12)
522:     .dc.w   __GMFontSize         ;; フォントサイズのセット
523:     addq.l  #4,sp
524:     rts
525:
526: ************************
527: *   ウインドウの終了処理
528: ************************
529:     .even
530: _SXterm:
531:     move.l  wPointer(a5),d0
532:     beq     __Exit
533:     move.l  d0,-(sp)
534: *       .dc.w   __WMDispose      ;; WMOpen で Window に0を指定した場合
535:     .dc.w   __WMClose            ;; WMOpen で Window に0以外を設定した場合
536:     addq.l  #4,sp
537: __Exit:
538:     .dc.w   __TSExit
539:
540: ****************
541: *   描画ルーチン
542: ****************
543:     .even
544: pset:
545:     movem.l d3/d4/a1,-(sp)
546:     lea     SetImg(pc),a1
547: psetCmn:
548:     move.w  16(sp),d3
549:     move.w  18(sp),d4
550:     add.w   d3,d3                ;; *2
551:     add.w   d3,d3                ;; *4
552:     move.w  d3,d0
553:     add.w   d3,d3                ;; *8
554:     add.w   d0,d3                ;; *12
555:     add.w   d4,d4
556:     add.w   d4,d4                ;; *4
557:     move.w  d4,d0
558:     add.w   d4,d4                ;; *8
559:     add.w   d0,d4                ;; *12
560:     swap    d4
561:     move.w  d3,d4
562:     move.l  d4,-(sp)
563:     pea     (a1)
564:     .dc.w   __GMPutRImg
565:     addq.l  #8,sp
566:     movem.l (sp)+,d3/d4/a1
567:     rts
568: preset:
569:     movem.l d3/d4/a1,-(sp)
570:     lea     ResetImg(pc),a1
571:     bra     psetCmn
572: *****************************************************************
573: * ここからはCコンパイラで作ったソースを変更したものです
574: *
575: * gcc -S -O -fomit-frame-pointer -fstrength-reduce -ffixed-a5
576: *  ¥-fcall-saved-d1 -fcall-saved-d2 -fcall-saved-a1 -fcall-sa
577: * ved-a2
578: *
579: * というオプションでコンパイルした後
580: * 変数をA5相対にするなどのちょっとした変更を加えた
581: *
582:     .even
583: life1:
584:     movem.l d1/d2/d3/a1/a2/a3/a4,-(sp)
585:     move.l  32(sp),d2
586:     moveq.l #-1,d3
587:     move.l  d3,_nv1(a5)
588:     move.w  #1,a2
589:     lea     _field(a5),a0
590:     move.l  d2,d0
591:     asl.l   #6,d0
592:     lea     1(a0,d0.l),a1
593:     lea     -63(a0,d0.l),a4
594:     lea     65(a0,d0.l),a3
595:     moveq.l #0,d0
596:     moveq.l #0,d1
597: L8:
598:     move.b  1(a1),d0
599:     move.b  -1(a1),d1
600:     move.w  d1,a0
601:     lea     (a0,d0.w),a0
602:     move.b  (a3),d0
603:     add.w   d0,a0
604:     move.b  (a4),d0
605:     add.w   d0,a0
606:     move.b  1(a3),d0
607:     add.w   d0,a0
608:     move.b  -1(a4),d0
609:     add.w   d0,a0
610:     move.b  1(a4),d0
611:     add.w   d0,a0
612:     move.b  -1(a3),d0
613:     add.w   d0,a0
614:     moveq.l #3,d3
615:     cmp.l   a0,d3
616:     beq     L5
617:     tst.b   (a1)
618:     beq     L4
619: L5:
620:     moveq.l #2,d3
621:     cmp.l   a0,d3
622:     bne     L6
623:     cmp.b   #1,(a1)
624:     beq     L4
625: L6:
626:     moveq.l #3,d3
627:     cmp.l   a0,d3
628:     bne     L7
629:     cmp.b   #1,(a1)
630:     beq     L4
631: L7:
632:     addq.l  #1,_nv1(a5)
633:     move.l  _vp1(a5),a0
634:     move.l  _nv1(a5),d0
635:     tst.b   (a1)
636:     seq     d1
637:     and.b   #1,d1
638:     move.b  d1,(a0,d0.l)
639:     move.l  _yp1(a5),a0
640:     move.w  a2,-(sp)
641:     move.b  1(sp),(a0,d0.l)
642:     addq.w  #2,sp
643: L4:
644:     addq.w  #1,a1
645:     addq.w  #1,a4
646:     addq.w  #1,a3
647:     addq.w  #1,a2
648:     moveq.l #62,d3
649:     cmp.l   a2,d3
650:     bge     L8
651:     subq.l  #1,d2
652:     moveq.l #0,d1
653:     cmp.l   _nv0(a5),d1
654:     bgt     L15
655:     move.l  d2,d0
656:     asl.l   #6,d0
657:     lea     _field(a5),a1
658:     add.l   d0,a1
659: L14:
660:     move.l  _yp0(a5),a0
661:     move.b  (a0,d1.l),d0
662:     ext.w   d0
663:     move.w  d0,a2
664:     move.l  _vp0(a5),a0
665:     move.b  (a0,d1.l),d0
666:     move.b  d0,(a2,a1.l)
667:     beq     L12
668:     move.w  a2,-(sp)
669:     move.w  d2,-(sp)
670:     bsr     pset
671:     bra     L16
672: L12:
673:     move.w  a2,-(sp)
674:     move.w  d2,-(sp)
675:     bsr     preset
676: L16:
677:     addq.w  #4,sp
678:     addq.l  #1,d1
679:     cmp.l   _nv0(a5),d1
```

▶ついにダンジョン・マスターを解いた。まあレッドドラゴンの強いこと強いこと。腕が吊ってしまった。さあ，あとはカオスの逆襲を待つばかり。でも，その前に謹賀新年 PRO-68Kの謎を探らなくては……。
池谷　尚紀(22)静岡県

```
680:     ble      L14
681: L15:
682:     move.l  _vp0(a5),d1
683:     move.l  _vp1(a5),_vp0(a5)
684:     move.l  d1,_vp1(a5)
685:     move.l  _yp0(a5),d1
686:     move.l  _yp1(a5),_yp0(a5)
687:     move.l  d1,_yp1(a5)
688:     move.l  _nv1(a5),_nv0(a5)
689:     movem.l (sp)+,d1/d2/d3/a1/a2/a3/a4
690:     rts
691: *
692: * 手でコーディングし直し（GNU　CCでは効率悪い）
693: *
694:     .even
695: life2:
696:     movem.l d2/d3/d4/a1,-(sp)
697:     move.w  #(64*DispHight-1),d2        ;; ループ回数
698:     moveq.l #0,d3                       ;; X座標
699:     moveq.l #0,d4                       ;; Y座標
700:     lea     _field(a5),a1
701: reDraw1Loop:
702:     tst.b   (a1)+
703:     beq     noReDraw
704:     move.w  d4,-(sp)
705:     move.w  d3,-(sp)
706:     bsr     pset
707:     addq.w  #4,sp
708: noReDraw:
709:     addq.w  #1,d4
710:     cmpi.w  #64,d4
711:     bcs     reDraw1
712:     addq.w  #1,d3
713:     move.w  #0,d4
714: reDraw1:
715:     dbra    d2,reDraw1Loop
716:     movem.l (sp)+,d2/d3/d4/a1
717:     rts
718:
719:     .even
720: initLife:
721:     movem.l d3/d4/d5/a3,-(sp)
722:     move.w  #1023,d1                        ;; ループ回数
723:     lea     _field(a5),a1
724: LL9:
725:     clr.l   (a1)+
726: .   dbra    d1,LL9
727: *
728:     moveq.l #0,d3
729:     lea     _field(a5),a3
730:     move.l  a3,d4
731:     lea     iniPat(pc),a3
732: LL10:
733:     move.b  (a3,d3.l),d0
734:     ext.w   d0
735:     move.w  d0,d1
736:     ext.l   d1
737:     move.b 1(a3,d3.l),d0
738:     ext.w   d0
739:     move.w  d0,a1
740:     addq.l  #2,d3
741:     cmp.w   #0,a1
742:     blt     LL11
743:     move.l  d1,d0
744:     asl.l   #6,d0
745:     move.l  d4,a0
746:     add.l   d0,a0
747:     move.b  #1,(a1,a0.l)
748:     move.w  a1,-(sp)
749:     move.w  d1,-(sp)
750:     bsr     pset
751:     addq.w  #4,sp
752:     bra LL10
753: LL11:
754: **
755:     move.l  #1,_xhold(a5)
756:     move.l  #-1,_nv0(a5)
757:     lea     _vs0(a5),a3
758:     move.l  a3,_vp0(a5)
759:     lea     _vs1(a5),a3
760:     move.l  a3,_vp1(a5)
761:     lea     _ys0(a5),a3
762:     move.l  a3,_yp0(a5)
763:     lea     _ys1(a5),a3
764:     move.l  a3,_yp1(a5)
765: **
766:     movem.l (sp)+,d3/d4/d5/a3
767:     rts
768: *
769: * ここまでがCコンパイラの出力
770: ****************************************************
771:
772: *
773: *         設定用パターン（青色）
774: *
775: SetImg:
776:     .dc.w   0,0,12,12               ;; bounds rectangle
777:     .dc.w   %11111111_11110000
778:     .dc.w   %11111111_11110000
779:     .dc.w   %11111111_00110000
780:     .dc.w   %11111111_00110000
781:     .dc.w   %11111111_00110000
782:     .dc.w   %11111111_11110000
783:     .dc.w   %11111111_11110000
784:     .dc.w   %11111111_11110000
785:     .dc.w   %11111111_11110000
786:     .dc.w   %11111111_11110000
787:     .dc.w   %11111111_11110000
788:     .dc.w   %11111111_11110000
789:
790:     .dc.w   %00001111_11000000
791:     .dc.w   %00111111_11110000
792:     .dc.w   %01111111_00110000
793:     .dc.w   %11100111_00110000
794:     .dc.w   %11100011_00110000
795:     .dc.w   %11000011_11110000
796:     .dc.w   %11000011_11110000
797:     .dc.w   %11100011_11100000
798:     .dc.w   %11100111_11000000
799:     .dc.w   %01111111_11000000
800:     .dc.w   %01111111_11000000
801:     .dc.w   %00011111_10000000
802:
803:     .dc.w   %00001111_11000000
804:     .dc.w   %00111111_11110000
805:     .dc.w   %01111111_11110000
806:     .dc.w   %11111111_11110000
807:     .dc.w   %11111111_11110000
808:     .dc.w   %11111111_11110000
809:     .dc.w   %11111111_11110000
810:     .dc.w   %11111111_11100000
811:     .dc.w   %11111111_11000000
812:     .dc.w   %01111111_11000000
813:     .dc.w   %01111111_11000000
814:     .dc.w   %00011111_10000000
815:
816:     .dc.w   %11111111_11110000
817:     .dc.w   %11111111_11110000
818:     .dc.w   %11111111_11110000
819:     .dc.w   %11111111_11110000
820:     .dc.w   %11111111_11110000
821:     .dc.w   %11111111_11110000
822:     .dc.w   %11111111_11110000
823:     .dc.w   %11111111_11110000
824:     .dc.w   %11111111_11110000
825:     .dc.w   %11111111_11110000
826:     .dc.w   %11111111_11110000
827:     .dc.w   %11111111_11110000
828: *
829: *         消去用パターン（灰色）
830: *
831: ResetImg:
832:     .dc.w   0,0,12,12               ;; bounds rectangle
833:     .dc.w   %11111111_11110000
834:     .dc.w   %11111111_11110000
835:     .dc.w   %11111111_11110000
836:     .dc.w   %11111111_11110000
837:     .dc.w   %11111111_11110000
838:     .dc.w   %11111111_11110000
839:     .dc.w   %11111111_11110000
840:     .dc.w   %11111111_11110000
841:     .dc.w   %11111111_11110000
842:     .dc.w   %11111111_11110000
843:     .dc.w   %11111111_11110000
844:     .dc.w   %11111111_11110000
845:
846:     .dc.w   %00000000_00000000
847:     .dc.w   %00000000_00000000
848:     .dc.w   %00000000_00000000
849:     .dc.w   %00000000_00000000
850:     .dc.w   %00000000_00000000
851:     .dc.w   %00000000_00000000
852:     .dc.w   %00000000_00000000
853:     .dc.w   %00000000_00000000
854:     .dc.w   %00000000_00000000
855:     .dc.w   %00000000_00000000
856:     .dc.w   %00000000_00000000
857:     .dc.w   %00000000_00000000
858:
859:     .dc.w   %00000000_00000000
860:     .dc.w   %00000000_00000000
861:     .dc.w   %00000000_00000000
862:     .dc.w   %00000000_00000000
863:     .dc.w   %00000000_00000000
864:     .dc.w   %00000000_00000000
865:     .dc.w   %00000000_00000000
866:     .dc.w   %00000000_00000000
867:     .dc.w   %00000000_00000000
868:     .dc.w   %00000000_00000000
869:     .dc.w   %00000000_00000000
870:     .dc.w   %00000000_00000000
871:
872:     .dc.w   %11111111_11110000
873:     .dc.w   %11111111_11110000
874:     .dc.w   %11111111_11110000
875:     .dc.w   %11111111_11110000
876:     .dc.w   %11111111_11110000
877:     .dc.w   %11111111_11110000
878:     .dc.w   %11111111_11110000
879:     .dc.w   %11111111_11110000
880:     .dc.w   %11111111_11110000
881:     .dc.w   %11111111_11110000
882:     .dc.w   %11111111_11110000
883:     .dc.w   %11111111_11110000
884: *
885: * ライフゲームの初期状態のパターン（座標）
886: *
887: iniPat:
888: *
889:     .dc.b   2,2,2,3,2,4                         ;; ブリンカー
890: *
891:     .dc.b   4,20,4,21,4,22,3,22,2,21            ;; グライダー
892: *
893:     .dc.b   6,30,6,31,5,31,7,31,5,32            ;; Rペントミノ
894: *
895:     .dc.b   6,3,6,4,6,5,7,3                     ;; πペントミノ
896:     .dc.b   7,5,8,3,8,5
897: *
898:     .dc.b   20,4,21,3,21,4,22,2                 ;; 世紀
899:     .dc.b   22,3,23,3
900: *
901:     .dc.b   19,13,20,13                         ;; ???
902:     .dc.b   19,14,20,14
903:     .dc.b   17,16,18,16
904:     .dc.b   19,16,20,16
905:     .dc.b   13,17,14,17
906:     .dc.b   16,17,21,17
907:     .dc.b   13,18,14,18
908:     .dc.b   16,18,17,18
909:     .dc.b   21,18,16,19
910:     .dc.b   19,19,21,19
911:     .dc.b   23,19,24,19
912:     .dc.b   16,20,18,20
913:     .dc.b   21,20,23,20
914:     .dc.b   24,20,17,21
915:     .dc.b   18,21,19,21
916:     .dc.b   20,21,17,23
917:     .dc.b   18,23,17,24
918:     .dc.b   18,24,-1,-1
919:
920:     .even
921: mess:
922:     .dc.b   'エラーでっせ。えらーいこっちゃ！',0
923:
924:     .even
925: wtitle:
926:     .dc.b   18,'ライフゲームだっ！'
927:
928:     .even
929: moduleEnd:
930:
931:     .offset 0
932: dataHead:
933:     ds.b    4096*2                          ;; スタック用
934: dataEnd:
935:     .text
936:     .end    execStart
```

```
¥
|-HUMAN.SYS
|-drv.x
|
|-sys
|    |-float2 + .x
|    |-RAMDISK.SYS
|-config.sys
|-autoexec.bat
|-start.R
|-COMMAND.X
|-qstart.LZH
|-card2.LZH
|-Michael.LZH
|-doctor2.LZH
|-vs2.LZH
|-dictools.LZH
|-Hash.LZH
|-SX.LZH
|-SX_sample.LZH
|-MUSICDRV.LZH
|-bin.LZH
|-Zs_EX.LZH
|-disk1.LZH
|-disk2.LZH
|-disk3.LZH
|-disk4.LZH

ArcFile:qstart.LZH
¥ (root)
          quickstart   <dir>
quickstart
          はじめに.doc
          disk1.bat
          disk2.bat
          disk3.bat
          disk4.bat

ArcFile:card2.LZH
¥ (root)
          card2   <dir>
card2
          CARD2.FNC
          CARD2.S
          CPUT.S
          CSET.S
          CSTAT.S
          CPALET.S
          CGET.S
          CBASE.S
          CARDLIB.A
          CARDDRV.x
          TR.DAT
          CDEF.x
          CDEF.bas
          CARDDRV.S
          klondike.bas

ArcFile:Michael.LZH
¥ (root)
          Michael   <dir>
Michael
          mi_menu.ws
          sure.ws
          mi_table.ws
          mi_mk_key.ws
          files.ws
          mi_para.ws
          mic.key
          mi_tool.ws
          mi_lines.ws
          org_mic.key
          org_mic.msk
          mic.msk
          mi_title.ws
          mi_char.ws
          mi_mode.ws
          mi_width.ws
          mi_calc.ws
          mi_exp.ws
          michael.x
          mi.bat
          work   <dir>
Michael¥work
          ヘキサr相関.mic
          HM20_TPD_data.mic
          ヘキサα相関.mic
          ZSM5_50TPD_DATA.mic

ArcFile:doctor2.LZH
¥ (root)
          doctor2   <dir>
doctor2
          doctor2.x
ArcFile:vs2.LZH
¥ (root)
          vs2   <dir>
vs2
          vs_cut.s
          vs_copy.s
          vs_init2.s
          vs_type.s
          vs_open.s
          cBase   <dir>
vs2¥cBase
          cbase.c
          dp.c
          tools.c
          cbase.x

ArcFile:dictools.LZH
¥ (root)
          dictools   <dir>
dictools
          makedic.x
          dumpdic.x
          addword.x
          msort.x
          convdic.x
          chdic.r
          delword.x
          split.x
          chkdic.x

ArcFile:Hash.LZH
¥ (root)
          Hash   <dir>
Hash
          hash.h
          differ.h
          frame.h
          vector.h
          hash.c
          hash_sub.c
          differ.c
          frame.c
          vector.c
          HASH.x

ArcFile:SX.LZH
¥ (root)
          SX   <dir>
SX
          README.DOC
          DOC   <dir>
          INCLUDE   <dir>
          LIB   <dir>
          REF   <dir>
          SAMPLE   <dir>
          TOOL   <dir>
SX¥DOC
          MEMORY.DOC
          EXCEP.DOC
          MOUSE.DOC
          ANIMA.DOC
          KEYBOARD.DOC
          KEY.DOC
          EVENT.DOC
          RESOURCE.DOC
          GRAPH.DOC
          WINDOW.DOC
          MENU.DOC
          CONTROL.DOC
          DIALOG.DOC
          TEXT.DOC
          TASK.DOC
SX¥INCLUDE
          SXLIB.H
          SXDEF.H
          SXCALL.EQU
          SXCALL.MAC
          DOSCALL.MAC
          IOCSCALL.MAC
SX¥LIB
          SXLIB.A
          __MAINR.O
SX¥REF
          MEMORY.REF
          EXCEP.REF
          MOUSE.REF
          ANIMA.REF
          KEYBOARD.REF
          KEY.REF
          EVENT.REF
          RESOURCE.REF
          GRAPH.REF
          WINDOW.REF
          MENU.REF
          CONTROL.REF
          DIALOG.REF
          TEXT.REF
          TASK.REF
          FUNC.LST
          CFORM.LST
SX¥SAMPLE
          GRSAMP.S
          TSSAMP.S
          TMSAMP.X
          DBLTIME.C
          DBLTIME2.C
          TSHELL.C
          GRSAMP.X
          TSSAMP.X
          TMSAMP.S
          DBLTIME.X
          DBLTIME2.X
          TSHELL.X
SX¥TOOL
          SXKERNEL.X
          SXWDB.X
          DB.X
          RLK.X

ArcFile:SX_sample.LZH
¥ (root)
          SX_sample   <dir>
SX_sample
          SXLIFE.S
          SXLIFE.X
          SXeyes.X
          15パズル.x
          パス名.x

ArcFile:MUSICDRV.LZH
¥ (root)
          MUSICDRV   <dir>
MUSICDRV
          MUSIC1.FNC
          MUSIC.DOC

ArcFile:bin.LZH
¥ (root)
          bin   <dir>
bin
          lh.doc

ArcFile:Zs_EX.LZH
¥ (root)
          Zs_EX   <dir>
Zs_EX
          mapicon.dat
          Zs_EX.x
          SOURCE   <dir>
Zs_EX¥SOURCE
          window.c
          effect.c
          picfiler.c
          mapping.c
          zs.c
          rfbuild.c
          startup.s
          xpic.s
          transfer.s
          rev.s
          compile.bat
          makefile
          rfbuild.x

ArcFile:disk1.LZH
¥ (root)
          autoexec.bat
          config.sys
          cut   <dir>
          quickstart   <dir>
cut
          kinga.cut
quickstart
          ForYou.PIC
          Weeds.pic
          itosiki.opm
          index90.rat
          Index90.fmt
          index90.dat
          Fuga.opm
          謹賀新年.doc
          tree_3d.pic
          sleepcat.pic

ArcFile:disk2.LZH
¥ (root)
          autoexec.bat
          config.sys
          CUT   <dir>
          QUICKSTART   <dir>
          font   <dir>
CUT
          caution.cut
          find.cut
          michael.cut
          option.cut
          zs.cut
QUICKSTART
          Zs_EX.DOC
          Michael.doc
          MUSICDRV.doc
          michael.bat
          readme.doc
          doctor.doc
font
          EGYPT.FON
          EGYPTB.FON
          EGYPTS.FON
          FTAIL1.FON
          NAMI.FON
          OPENG.FON
          ROMAN3.FON
          ROMAN4.FON
          KUMO1.FON
          KUMO2.FON
          KUMO3.FON
          POMP.FON
          POMPB.FON
          POMPD.FON
          POMPH.FON
          POMPW.FON
          BISCUIT.FON

ArcFile:disk3.LZH
¥ (root)
          autoexec.bat
          config.sys
          cut   <dir>
          quickstart   <dir>
cut
          card.cut
          hash.cut
quickstart
          klondike.x
          RING.Hsh
          PYRAMID.Hsh
          BASLLS.Hsh
          card2.doc
          Klondike.doc
          hash.doc

ArcFile:disk4.LZH
¥ (root)
          autoexec.bat
          CUT   <dir>
          QUICKSTART   <dir>
CUT
          sx.cut
QUICKSTART
          sx.doc
```

謹賀新年 PRO-68K

Oh!X史上2番目の付録ディスクをお届けしよう。ちなみに左が今回のディスクの内容を一覧としたもの，だ。
今回のディスクにはいくつかの目標があった。そのうち，ひとつはシステムを入れ，直接起動可能にすること。ディスク容量を削ることになっても，だ。さらに，ソースプログラムをできるだけ収録すること，できるだけ掲載プログラムを入れないこと，できるだけ使いやすくすること……。これらがどの程度達成されたか，その目標が正しい目標であったかについて，その評価はこのディスクを手にしたあなたに委ねることにする。
ディスク容量の関係から削除せざるをえなかったものもある。時間的に十分でなかった面もある。それでも,「ディスクでしかできないこと」を，実現したと信じたい。

CONTENTS

付録ディスク利用の手引き

編集部

お待たせしました。ついにディスク第2弾です。多くの方から期待のお手紙をいただきました。ありがとうございます。今回は前回のディスクでの反省点もふまえて，初心者にもわかりやすい構成にしてみました。ではまず付録ディスクの取り扱い方，圧縮されたファイルの取り出し方などについて解説しましょう。なお，起動時はRAMディスクがAドライブに設定されていますので注意してください。

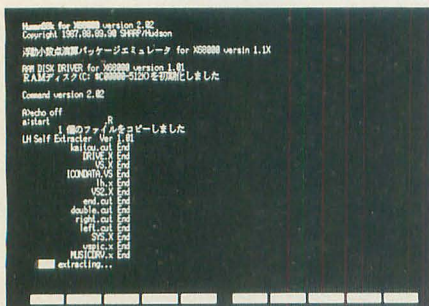
みなさま，あけましておめでとうございます。

予告どおり今月はディスクつきです。1990年6月のような大物はありませんが，それとはまた違ったアプローチからディスクを構成してみました。

ファイルの圧縮にはLH.X ver.1.01αを使用しています。これは吉崎栄泰氏の原作，山本浩一氏の移植によるものです。ディレクトリごとの圧縮など，今回のディスク作成では大活躍しています。そのため，「やったー！　1月号は特別付録だー。生ディスク3枚用意して待ってますよん」という宮城県の相沢さんには申しわけないのですが，生ディスクは4枚必要となってしまいました。

今回のディスクおよびそれを展開したものすべてがシステムつきですのでそのまま起動可能です。

X68000には発売当初よりビジュアルシェルが搭載されていたことはご存じのとおりですが，クオリティグラフィックやマルチメディア，高速性など真の意味での汎用性を志向したユーザーコンセプトからは，私たち自身もの足りなさを禁じ得なかったことも事実です。そこでVS2.Xを開発しました。流通する多くのコンピュータが16色表示という環境のなかで，高解像度65536色グラフィックを実現しています。

起動のようす

今回のディスクはすべてVS2.X上から操作します。

ディスク内のほとんどのファイルは圧縮されています。使用する際には所定の解凍作業が必要です。受験生は自己の責任において解凍するようにしてください。

またSX-WINDOW関係はシャープ提供の開発者向け資料を編集部で独自にまとめたものです。こういったものは世の常として，使用法や内容に関する質問をシャープに問い合わせることは絶対にしないでください。編集部に問い合わせることもご遠慮ください。質問されても回答できません。

新年早々にもソフトバンクからSX-WINDOWの解析本が発売される予定ですので，それらを参考にしてください。どうしてもわからない部分はたぶんこちらでもわかりません。サンプルプログラムやシステムプログラムを読んでください。コードがすべてを語ってくれるはずです。

また，「Oh!Xでは半年に1回ディスクをつけるんだ」と早合点しないように。

ディスクの使い方

具体的に付録ディスクを起動してからの作業を列記してみましょう。

1)　プロテクトシールを貼る

磁性面には貼らないでください。

これが起動画面

2)　ディスクを起動する

起動すると勝手に延々と必要なファイルを展開します。とりあえず無視してください。やがてビジュアルシェルが立ち上がります。起動時はできるだけシフトキーを押しながらリセットするようにしてください(RAMディスクの初期化)。うまくいかない場合は電源を落とします。X68000のRAMはタフですから電源を落として10秒以上ほうっておいてください。

3)　「はじめに.DOC」のアイコンをダブルクリック

説明が出ます。

4)　新しいディスクをドライブ1に入れる

必要ならフォーマットします。

5)　バックアップを取る

展開作業に入る前にディスクをバックアップしてください。BドライブのディスクアイコンをつかまえてCディスクのところに重ねます。完璧に重ねようとすると失敗します。あくまでもマウスカーソルの先がCドライブのアイコンを指すように重ねてください。

6)　バックアップディスクを取り出す

Cドライブのアイコン上で右クリックしメニューから「close/eject」を選んで指を離します。

7)　新しいディスクを入れる

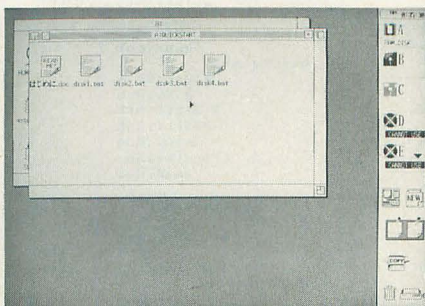
必要ならフォーマットし，すでになにか入っているディスクの場合もディスクアイコンのところからフォーマットを選んでください。

8)　展開作業に入る

DISK1〜4.BATをダブルクリックします。なお，展開したディスクはすべてIPL起動可能です。

例によって，現在までに判明した今回のディスクの問題点を挙げておきましょう。

●1Mバイトでは動かないものがある

Michaelはビジュアルシェルからでは起

動できません。Zs_EXはできれば4Mバイト以上で使ってください。
●Zs_EXのアイコンが違う
使う分には関係ないのですが，MAP時のアイコンはZ'sTRIPHONYのものです。新たに作り直したのですが，手違いで開発中の暫定版(?)が入ってしまいました。

●ディスク4のREADME.DOCの内容が違う
容量の都合で一部のサンプルはディスク3に移動しました。とけい.Xは収録されていません。理由は聞かないでください。
●SXLIFE.Xが全クローズで消えない
個別に消してください。

●ボリューム名をつけそこなっている
最後につけた部分でした。
●一部のドキュメントに誤字がある
余力があれば直す予定でした。

＊　＊　＊

細かい部分は解凍したディスクを実際に起動して確認してみてください。

謹賀新年PRO-68Kを使うには，あるいはX68000 SUPERユーザーに贈るビジュアルシェルの使い方

Nakano Shuichi
中野 修一

発売当初のX68000を強烈に印象づけたもの，ひとつはいまだ色褪せない名作「グラディウス」，そしていまひとつはビジュアルシェルことVS.Xでしょう。
さて，今回のオマケディスクはビジュアルシェル上で動作します。前回手落ちがあった展開作業も，ディスクさえ入れてやれば，あとはアイコンをダブルクリックするだけで自動的に完了します。
Human68kやCOMMAND.Xのバージョンの違いによる誤動作を防ぐため，最新のHuman68k ver.2.02を搭載しました。これで初期型X68000を買って4年間ver.1.0のHuman68kしか使っていなかった方も安心です。
さらに「OPMDRV.Xはどうやって手に入れるんですか？」という方のために，豊富な実績を誇るOPMDRV.X ver.1.01を，そのほか，貴重な磁性面を割いてプログラムの動作に必要なデバイスドライバをすべて同梱しました。
そして，「せっかくのディスクをフォーマットしちゃったんですが……」という悲劇を繰り返さないためにも，猿でもわかるビジュアルシェルが搭載されたわけです。
一説によると「ユーザーフレンドリ」ともいいますが，「全部面倒みなけりゃ，どうせなにもできねえだろ」と，いうことですから，精鋭を集めたOh!Xの読者としては馬鹿にされていると憤慨するほうが健全です。
当然SX-WINDOWが注目されるところですが，メインメモリ1Mバイトで動作させるため，VS.Xが日の目を見ることになったのです。ディスクに入っているVS.Xはまったく従来どおりのものですが，VS.XはX68000SUPERなどのSCSIデバイスに対応していないなどの問題があるため，独自に拡張を加えたVS2.Xも用意しました。
今後の動向は流動的ですが，少なくとも次にディスクをつけるときにも同様の構成をとることになるでしょう。特にX68000SUPERユーザーの方，ビジュアルシェルの操作に慣れておいてください。

ビジュアルシェルの基本

まず，基本動作から，
●ディスクのフォーマット
生ディスクを挿入すると，フォーマットするかどうか聞いてきます。
●ディレクトリの移動
ディレクトリアイコンをダブルクリックすればそのディレクトリのウィンドウが開きます。通常はQUICKSTARTのディレクトリが最初に開きますので，ルートに戻りたいときはドライブアイコンをダブルクリックします。
●ファイルのコピー
SX-WINDOWと違って，ウィンドウ上のアイコン移動はすべてムーブ(要するに元の場所からなくなる)となるので，コピーしたい場合は一度コピー機のアイコンのところまで持っていってください。コピー機から飛び出したアイコンをつかんで持っていけばコピーが実行されます。
●ディスクのイジェクト
ディスクアイコンのところで右クリックしイジェクトを選んでください。
●ディスクのコピー
ディスクアイコンをひっつかまえて，目的のディスクの上に重ねてください。ハードディスクとフロッピーディスクなど，違う種類のメディアはディスクコピーできません。同じメディアでも容量が違うとディスクコピーできません。
フロッピーディスクがEドライブ以降に割り当てられているときは，画面上に1個のディスクアイコンしか出せません。そんなときは，いったんコピー機のところへアイコンを置いてからディスクを切り替えてください。
●終了
なにも選んでない状態で右クリック，QUITを選択します。

応用

ここまでは基本。単に付録ディスクを使うためならこれで十分です。通常のシステムディスクに入っている全然使いこなされていないVS.Xを見てビジュアルシェルはなにもできないと思っている人がいるともったいないのでもう少し解説します。
さらに使いこなすためのキーとなるのはアイコンです。
アイコンはただ並んでいるだけではありません。アイコンのそれぞれはディスク上のプログラム，データを表します。それぞれなんらかの目的のもとに作成されたもので，呼び出されれば所定の機能を果たします。普段は小さくて必要なときに大きくなるカプセル怪獣のようなものです。
そして，ダブルクリックはその眠りを醒まします。では，眠りから醒めたファイルはどうなるべきでしょうか？
通常，さまざまなファイル，BASICのテキストファイルとか，文書ファイル，音楽データなど，プログラムでは違うものとして扱いますが，ディスク上ではなんの違いもないデジタル情報，ビット列にすぎません。これらが本来どのように扱われるべきかというのは，決まっているようで決まってないのです。
もしかして，*.X形式の実行ファイルを65536色のグラフィックと見なして表示したいとか，グラフィックツールで描いた絵をADPCMで鳴らしたい……と思う方もいるかもしれません。可能です。データがどのようにふるまうべきかというのはユーザーがすべて指定できるのです(対応するプログラムがあれば，だが)。
これを指定するのがアイコンメンテナンスです。実行ファイルとパラメータを指定することができます。起動したファイル名は自動的に渡されますから，

起動ファイル
ファイル名
パラメータ

と，ひととおりの指定が可能です。付属ディスクではおおまかな指定がしてありますが，コマンドシェルが使える人なら独自のシステムを作ることも簡単でしょう。
通常の実行ファイルは無指定でも，そのまま実行されます。データファイルの場合はなんらかの実行ファイルを指定します。なにも指定しないと「ファイルは実行できません」エラーになります（VS2はこの部分を取り出して拡張している)。
このとき，起動ファイルのパス指定はどうすればよいのでしょうか。基本はあくまでフルパス指定です。が，ちゃんとそのアイコンがあった場所がカレントになりますのでドライブを省略したりすると各データファイルの位置に依存した指定となります。
ですから，おまけディスクの*.PICのファイルはアイコンのオプションを，

=¥BIN¥VSPIC %

から，

=A:¥BIN¥VSPIC %

に変更しましょう。

＊　＊　＊

時代はもうSX-WINDOWに向かって流れているのかもしれません。そのSX-WINDOWは当然，VS.Xを超えるものでなければなりません。しかし，発表以来なんの進化もさせてもらえなかったVS.Xを超えてもあまり意味はないでしょう。VS.Xの延長で可能であっただろう部分を含め，真っ向から超えていってほしい，そう思います。

Shell

VS2.Xの使い方

ビジュアルシェルの基本的な使い方がわかったところで，今回拡張された部分について見ていきましょう。シングルウィンドウとあまり志は高くありませんが，VS.2はもっとも簡単な方法で最大限の効果をあげています。皆さんのご意味をお聞かせください。

Kageyama Hiroaki
影山 裕昭

これから紹介するVS2.XはVS（ビジュアルシェル）にちょっとばかし便利な機能を追加するプログラムです。このプログラムはOh!Xのオマケディスク用のシェルとして企画/開発されました。

VS2.Xの起動はコマンドラインから，

A>VS2

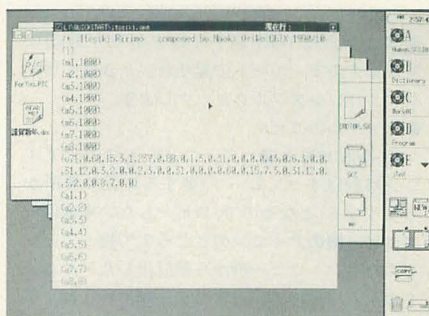
で行います。パスの通ったディレクトリにバージョン1.12のVS.Xがあると，これから紹介する拡張機能が使えるようになります。追加された機能は，

ファイルの内容表示：>TYPE
ファイルのコピー　：>COPY
外部ファイルの実行：=

の3つです。これらはすべてアイコンメンテナンスのパラメータで設定します。

ファイル内容の表示

VSでは*.Xファイルをダブルクリックで実行することができました。どうせなら，ダブルクリックでファイル内容の確認もできればもっと便利だろうということで作ったのが>TYPEコマンドです。ここでは拡張子がDOCのファイルをダブルクリックしたときに，ファイル内容を表示するようにしてみましょう。

まず最初にVSのアイコンメンテナンスを選択します。わからない人はX68000の取扱説明書，2.6デスクアクセサリアイコンの（8）アイコンメンテナンスの使い方，を参照してください。ウィンドウが開いたら，

アイコン名　：*.DOC
実行ファイル：
パラメータ　：>TYPE %

と書きます。コマンドは大文字でも小文字でも構いませんが，すべて半角文字で書いてください（以下紹介するコマンドはすべて半角で書きます）。これで準備完了です。アイコンメンテナンスを終了して，拡張子がDOCのファイルをダブルクリックしてみてください。普通なら「このファイルは実行できません」となるのですが，パラメータに>TYPE %を指定してあるので，ウィンドウがオープンしてファイル内容が表示されます。ここでのキー操作をまとめておきます。

普通にファイルを表示

80桁で表示した例

F1：先頭行に移動
F2：最終行に移動
↓：正スクロール
↑：逆スクロール

ほかにロールアップ，ロールダウンが使えます。また電脳倶楽部のようにマウスの左クリックで正スクロール，右クリックで逆スクロール，両クリックでウィンドウをクローズすることができます。画面上ではウィンドウの左上のマークをクリックするとウィンドウをクローズします。また右上のマークをクリックするとウィンドウを大きく開きます。いい忘れましたが，初めからウィンドウを大きく開きたいときはパラメータに，

>TYPE −80 %

と指定してください。80は80桁という意味です（将来は可変にするつもりだが，現バージョンでは80以外はエラーとして扱っています)。これはトグルスイッチになっているので，続けてクリックすると元の大きさに戻り，さらにクリックするとまた大きく……，とウィンドウのサイズを交互に変更することができます。

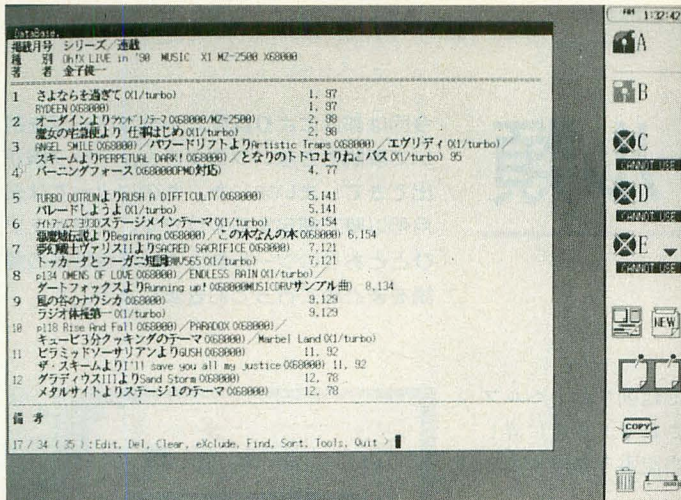
さらに，このタイプコマンドでは電脳倶楽部でお馴染みのカットファイルを表示す

カットファイルも表示できる

VS2上のプログラム

VS2はマルチタスクでもマルチウィンドウでもない。基本的にポップアップウィンドウひとつしか扱わない。わざわざ，このうえでプログラムを走らせるより，ふつうに起動したほうが面倒がない。あえてやっている理由は「雰囲気がよい」の一点に集約される。

よって，VS2上のプログラムはもっぱら，ウィンドウの中で動いているふりをすることが要求される。その代わり，ハードウェア資源のうち，スプライトとテキストプレーンの0，1を除いたすべてが使える。プログラム作成はとても楽である。

今後の方向として，常駐終了させたタスクを切り替えたり，コンソールを開いたりといったことも理論上は不可能ではない。が，プログラムの性格上，1Mバイトで動いて，十分に小さいものが要求されているのであまり期待はしないように。

外部コマンドとして用意されたcbase

PICファイルの表示例

ることもできます（ただし白黒が反転している）。

ファイルのコピー

これまでに本誌の音楽特集などでVSからダブルクリックでOPMファイルを演奏する方法を紹介してきました。あれを試した人はわかると思いますが，見ていてあまり美しいものではありません。それはファイルのコピーにCOMMAND.Xを使う必要があったからです。そこでコピーコマンドを追加しました。コマンドは>TYPEと同じようにパラメータに指定します。

アイコン名　：*.OPM
実行ファイル：
パラメータ　：>COPY % OPM

とパラメータに書いておくだけで，VSを抜けずにOPMドライバにファイルをコピーできるようになります。もちろんOPMドライバが登録されていなければ演奏されません（あたりまえですね。未登録の場合はカレントドライブにOPMという名前のファイルが作られます）。

ほかにも，

>COPY % PCM
>COPY % @IOCS

といったようなこともできます。各自工夫してみてください。

外部コマンドの実行

最後に紹介する＝コマンドは外部プログラムの実行をするものです。普通の実行と違い，VSの画面が表示された状態のままプログラムが実行されます。ですから指定する外部プログラムはVS用に作られたものでないと，実行終了後に画面が崩れることもあります。

今回はPICファイルを表示するVSPIC.Xを丹氏が，カード型データベースCBASE.Xを泉氏が作成しています。

CBASE.Xの解説は泉氏の記事を読んでもらうとして，VSPIC.Xについて触れておきましょう。付録ディスクにはVSPIC.Xが収録されています。このプログラムを使えばVSの画面上でPICファイルを見ることができます。

アイコン名　：*.PIC
実行ファイル：
パラメータ　：=A:¥BIN¥VSPIC %

のようにします。実行ファイルにはなにも指定せず，パラメータで＝に続けて実行ファイルを指定してください。設定すればメーカー標準のGL3ファイルもロードできますが，今回のディスクに収録されているアイコン設定ではサポートしていません。

また，左記のように実行ファイルは基本的にフルパスで指定してください。表示された画像を消すときはマウスを両方クリックするか，ESCを押しましょう。

最後に

今回収録したVS2.Xは改善の余地がかなりあり，自分自身では不満な点も多くあります。開発期間が短く，急いで作って中途半端なものにしたくなかったので，今回はあえて評価版というかたちにさせてもらいました。ご意見，ご要望があれば編集部までお寄せください。

次の機会にはバージョンアップしたものを紹介する予定です。

VS2.Xの高速化

今回のディスクではIOCS.Xを組み込んであるので，アイコンの表示などが高速化されているが，それ以上の高速化の手段もないではない。

満開製作所発行の電脳倶楽部vol.21で発表されたMOUSE.Xを組み込むとマウスが軽くなる。特にマウスカーソルのアニメーションなどが高速化され，結果としてウィンドウのクローズなどが速くなる。

また，通常CONFIG.SYSで設定されている，「VERIFY ON」の設定をOFFに変えるとディスク操作が大幅に高速化される。これはディスクのエラーをチェックする部分なので，常識的にいって，不用意に外すべきものではない。これまで，VERIFY OFFで作業している人は少ないと思う。いままで使っていて，ディスク速度で不満を持ったことがないならあえてOFFにする必要はない，ともいえる。逆に，これまでディスクエラーなど出たことがないし……という場合ならOFFにしといてもいいんじゃないか，ともいえる。各自の責任において判断してほしい。

そのほか，TYPEのスクロールは山ほどウエイトが入っているので好みで調整すること。

VSPIC

柳沢明氏によるPIC.Rのルーチンを使ってビジュアルシェル上に画像を表示するもの。すでにご存じの方も多いと思うが，X68000は768×512モードでも制限付きながら65536色表示することが可能である。これはビデオコントローラを直接操作することにより実現される。

問題となる制限は「512×512ドット分しか表示できない」というものだ。今回は512ドット分以外のところをマスクすることで1枚だけグラフィックウィンドウを確保している。もともとVS2はシングルウィンドウのシステムだから特に問題はない。

このモードではドット比が1：1に近く，また，画素も緻密である。大きさを除けば，X68000でもっともグラフィックの美しいモードといえる。今回はPICファイル用にドット比を補正して表示している。よほどのことがないかぎり，不自然さはないと思う。

etc. システムおよびツール類

今回は都合により誌面で十分なページを割いて解説できないプログラム，データ群が出てきてしまいました。ものによっては来月号以降で補足する予定です。とりあえず，ひととおりのツール群について最低限の解説をまとめて行っておきましょう。

編集部

CBASE.X (DISK1)

X-BASIC調理実習で泉大介氏が作成したカード型データベースをC言語にコンバートしてチューンアップしたもの。VS2.X上で動作する2番目のプログラムだ(1番目はVSPIC.X)。といっても構造はふつうのプログラムと変わりない。主に画面表示部分を変更，呼び出され方でCOMMAND.XからでもVS2.Xからでも同様に動作する。

テキストデータの取り込み，書き出しや複数項の一括検索など，実用に耐える仕様に強化されている。扱えるデータ数はまだ万全とはいえないが，小規模の用途でなら十分に使える。

詳しくは今月のX-BASIC調理実習を参照のこと。

INDEX90 (DISK1)

昨年（1990年）1年間のOh!X掲載記事をカード型データベースに入力したもの……のはずだったが，残念ながら，データ入力は不完全である。特集，連載/シリーズもの関係はほぼ実用レベルのデータが揃っているが，単発記事は全滅に近い。データを補完するもよし，CBASE.X用のサンプルデータとして使ってほしい。

DICTOOL (DISK3)

1990年9，10月号で村田敏幸氏が発表したASK68K用の辞書メンテナンスツール群。辞書ファイルから特定の品詞だけテキストデータにしたり，編集したテキストデータから新しい辞書を構成したり，X68000やPC-9801用に発売されている日本語フロントプロセッサ用の辞書をASK68K用にコンバートしたり，といった日本語入力環境を改善するためのもの。

簡単な使用法ならば，それぞれのプログラムをパラメータなしで起動するとヘルプメッセージとして表示される。さらに詳しい使い方は掲載号の該当記事を参照すること。

FONT (DISK1)

IOCS.Xのフォント切り替えのサンプルとして，過去に電脳倶楽部に掲載された平木敬太郎氏の作成による8×16ドットの半角英数字フォントを収録したもの。

IOCSのフォント切り替えにはIOCS.XがCONFIG.SYSによってデバイスドライバとして組み込んであることが前提条件となる。

方法はコマンドライン上からは，

A>COPY NAMI.FON @IOCS

のように，システムファイル名“@IOCS”にデータをコピーするだけだ。付録ディスクによって作成されたVS2.X上でなら，フォントファイルのアイコンをダブルクリックするだけでよい。今回は変更用のフォントしか用意していないので，切り替えたフォントを元に戻すにはリセットするしかない（IOCS.Xを解除すれば別だが）。注意してほしい。

HASH.X (DISK2)

一圓亨氏による半影つきレイトレーサ。通常，レイトレーシングを始め，コンピュータグラフィックでは難しいとされる柔らかな影を実現したレンダリングシステムだ。これは，HAlf SHadow ray tracingシステム，HASHというプログラム名でディスクに収録されている。

CBASE. X

切り替わったフォント

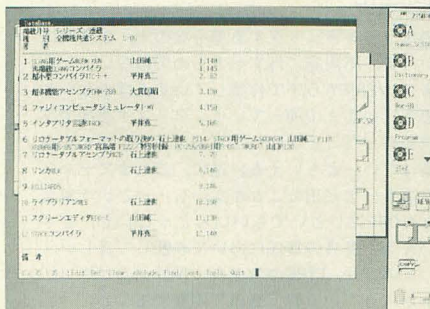
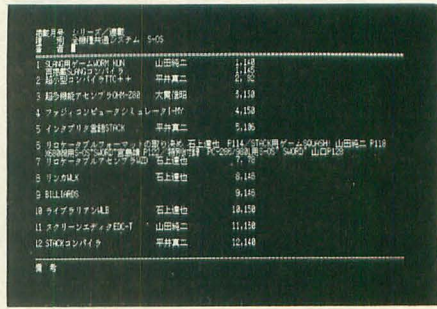
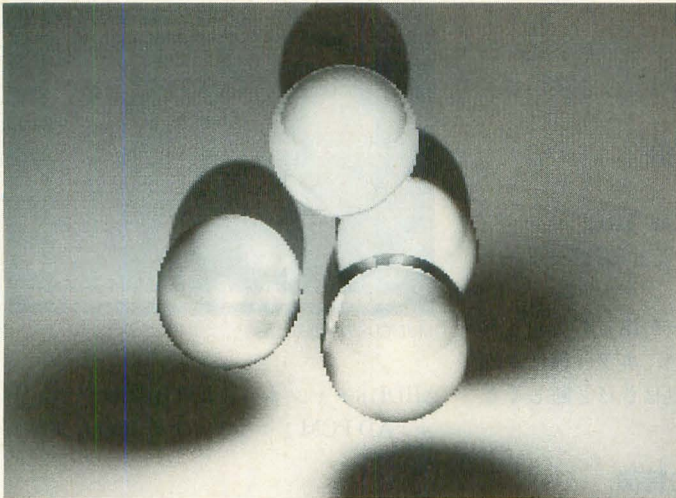
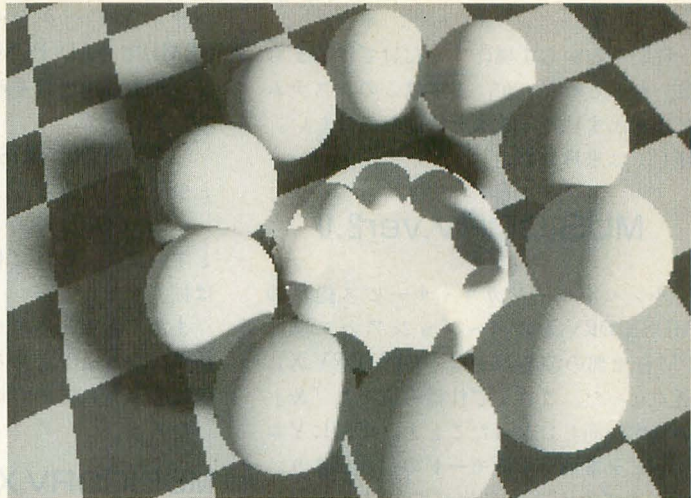
残念ながら，アルゴリズムなどの詳しい解説は来月に譲ることにして，ここでは，基本的な使い方とサンプルについて解説する。

まず，使用法。コマンドラインから，

A>HASH データファイル

で起動する。途中で終了チェックをしていないので，どうしても計算を途中でやめたいときはインタラプト＋Aを押す。VS2.X上からは，単にデータファイルのアイコンをクリックするだけでとりあえず実行は始まる。

データファイルはED.Xなどのテキストエディタで作成する。

このシステムで扱える物体は球のみで論理演算はない。レイトレーシングの経験がある人ならば，データファイルをのぞき見

RING

BALLS

るだけで，独自のデータファイルを作成できるだろう。

最初にあるS項はシステム設定で画面の解像度や色数，ディザ，表示範囲，解像度，反射回数，半影をつける際の解像度を表す。そのほか視点設定，光源設定，アトリビュート設定，背景設定などは注釈を見ればだいたいの見当がつくはずだ。

●PYRAMID

光源は，画面左上に見えている無限遠光源とピラミッドの頂点の球状光源，それに視点の無限後方からピラミッドを照らすものの3つ。計算時間は，mesh＝40で10.97時間，mesh＝1(通常の影処理)で6.87時間。この場合，通常の影処理の場合に比べて1.6倍程度の時間で半影を生成している。なお，ここに記す計算時間はGNU－C，FLOAT3＋.X，数値演算プロセッサボード→256×256ピクセルによるものであるが，計測によると'FLOAT2＋.X'でも時間的にはさほど変わらないようである。原因は，半影処理にほとんど実数計算を使わなかったことにあるとも思えるが，数値演算プロセッサが遅すぎるのではないだろうか。

●BALLS

4つの球を正四面体の頂点に配置。白い球が光源で，非常に半径が大きいため他の3つの球のぼやけた影を広く地面に落としている。球の向こう側に見えるシャープな影は，見掛けの大きさが非常に小さな無限遠光源による影である。計算時間は，mesh＝40で3.16時間。

●RING

HSV色座標系で求めた10色の球を円形に配置。地面に一部だけ顔を出した球体に映り込んでいる。地面は，白と緑のチェック模様。計算時間は，mesh＝20,86×86ピクセルで12分。

HASHのデータサンプル

```
================  PYRAMID.Hsh  =================
 1: *  pyramid  *
 2:
 3: *  wid hgt bit dth    x1 y1  x2  y2     dx dy  max  mesh  *
 4: s  512 512  16  0     0  0  511 511     2  2   3   40
 5:
 6: *  画面幅/2  上方向    参照点       視点  *
 7: v     80    0 0 1   0 0 10    -130 -75 35
 8:
 9: *  par      x   y   z       r    light intensity  *
10: l  -1     12. 12. 2.       3      1.0 1.0 1.0
11: l  -1     -3. -3. 1.    .2666     0.7 0.7 0.7
12:
13: *  No    red  grn  blu  ref high blinn  *
14: a   0  1.00 .875 .250  .30  0.1  20
15: a   1  .875 1.00 .250  .30  0.1  20
16:
17: *  chek at1 at2  half dift amb        sky color  *
18: g   50.  0   1    900.  .8  0.    .404 .242 .646
19:
20: *  No    red  grn  blu  ref high blinn  *
21: a   2  1.00 .600 .600  .20  0.    0
22: *  atr     x   y   z     r    light intensity  *
23: l   2     0   0  40     5      1.0 1.0 1.0
24:
25: *  No    red  grn  blu  ref high blinn  *
26: a   3  1.00 .600 .600  .20  4.    64
27:
28: *  atr     x    y    z      r  *
29: p   3   -10   10   30     5
30: p   3    10   10   30     5
31: p   3   -10  -10   30     5
32: p   3    10  -10   30     5
33:
34: p   3    20   20   20     5
35: p   3     0   20   20     5
36: p   3   -20   20   20     5
37: p   3    20    0   20     5
38: p   3     0    0   20     5
39: p   3   -20    0   20     5
40: p   3    20  -20   20     5
41: p   3     0  -20   20     5
42: p   3   -20  -20   20     5
43:
44: p   3   -30   30   10     5
45: p   3   -10   30   10     5
46: p   3    10   30   10     5
47: p   3    30   30   10     5
48: p   3   -30   10   10     5
49: p   3   -10   10   10     5
50: p   3    10   10   10     5
51: p   3    30   10   10     5
52: p   3   -30  -10   10     5
53: p   3   -10  -10   10     5
54: p   3    10  -10   10     5
55: p   3    30  -10   10     5
56: p   3   -30  -30   10     5
57: p   3   -10  -30   10     5
58: p   3    10  -30   10     5
59: p   3    30  -30   10     5
60:
61: e
```

＊　＊　＊

理屈もわからず操作するだけではつまらないかもしれないが，まずはこのシステムが作り出す独特の映像を堪能してほしい。詳しくは来月号で。

MUSICDRV.ver2.0

サン・ミュージカル・サービス提供のMUSICDRV.Xのバージョンアップ版。FM音源部の演奏がかなりOPMDRV.Xに近くなった。これまで仕様の違った「&」の動作が同じになったことと，OPMにYコマンドを素通しするモードの追加により，これまでのBASICに比べてもほとんど違和感のないコーディングが可能。MIDI部分の高性能はそのままに和音関係ほかのバグもなくなり，MIDI制御には絶対の強さを誇る。

おまけにAD PCM関係ではOPMDコンパチのYデータを受け付ける。加えてOPMD用のAD PCMコンフィギュレーションがある場合はそれがそのまま使用できる。これまで，

A>OPMD BOS.CNF

とやっていた方なら，

A>COPY BOS.CNF MIDI

とすればよいわけだ（ただしPCMバッファは拡大すること）。

まだ，ときおりエラーを出しておかしな演奏になることもあるが，エラー箇所は的確に表示されるので対処はさほど難しくないだろう。

MUSICDRV.Xの特徴

さて，MUSICDRV.Xといっても全然わからない方もいると思う。ここでその特徴をひととおりながめてみよう。

●MIDI＋FM音源＋AD PCM

まず，音源ならなんにでも対応している。

MUSICDRVの組み込み

MIDI16チャンネル，FM音源8チャンネル，AD PCM 1チャンネルが同時に演奏可能。

●和音指定

16チャンネルのシーケンスができる，とはいっても，本来1チャンネルで多くの音を出力できるMIDI楽器に比べ，OPMDRVなどは1トラック1音が基本となってる（FM音源なら当然のことだが）。このまま

カードゲーム　KLONDIKE（DISK2）

柴田雅隆

KLONDIKEはソリティア，つまり一人遊びの一種です。一見退屈そうなトランプ遊びですが，やってみると不思議不思議，ついつい時のたつのを忘れて没頭してしまうことでしょう。このプログラムは福岡県の柴田雅隆さんによる投稿作品です。

●

DISK 2を立ち上げ，KLONDIKE.Xのアイコンをダブルクリックしてください。起動すると，写真1のようにカードが並べられます。7列の場札とその上部に4つの黄色い枠があるのがおわかりでしょう。ゲームの目的はこの枠の位置にすべてのカードを4つのスート（スペードとかハートとかのマークのこと）ごとに1から13(KING)の順で積み上げるというもので，これをファウンデーションといいます。写真3が完成の状態です。

ではその手順ですが，場札の各列いちばん上の表向きになったカードに注目してください。写真1の場合，さっそくクラブのAが出ていますので上部の枠のどれかに上げてしまいましょう。以後，クラブの2，3，4……と同じスートのカードが出るたびに重ねていくことができます。場札からカードが移動すると，その下にあったカードが表向きとなります。

さて，場札のなかでも表を向いたカードは一定のルールでカードを移動でき，ここでは，数字が小さくなる順にしかも赤札と黒札が交互になるように重ねることができます。たとえば，写真1の場合ですと，左から1列目のハートの9は5列目のスペードの10の上に重ねることができます。この場合ですとハートの9が移動した跡は空きスペースとなりますが，KINGがあればここに移動することができます。

さらに，重なったカードはまとめて移動することも可能です。たとえば，黒7赤6黒5と重なった3枚のカードは，赤の8があればその上に移動できるわけです。

操作は簡単，マウスのカーソルを移動したカードの上に合わせ，左ボタンを押しながら動かすと目的の位置までカードを移動することができます。重なったカードをまとめて移動するには表を向いたいちばん下のカードにカーソルを合わせ右ボタンでまとめて移動します。

また，画面の右上にはストックがあり，ここを左クリックすると，3枚ずつカードがオープンになり，ここからもカードを移動することができます。移動可能なカードがなくなったらストックをクリックしてください。ストックが1周したらNEXTと表示された部分をクリックします。2周目は2枚ずつ，3周目は1枚ずつオープンしていきます。

ストックが3周すると，ENDマークが表示されます。あとは表向きのカードを慎重に見比べて可能なかぎり移動を試みてください。移動可能なカードがなくなったらゲームオーバーです（写真2）。

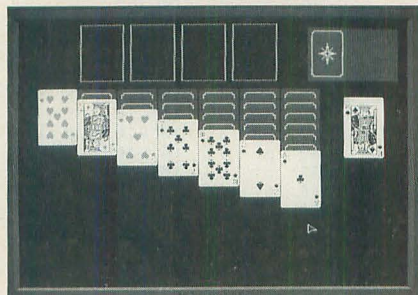



写真1　初期状態

なお，いったんファウンデーションの位置に上げたカードでも重なりの条件を満たすかぎり，場札の位置に戻すことも可能です。うまく利用すれば上がる可能性が高くなるでしょう。

注意：このプログラムはCARD2.FNCおよびCARDDRV.Xを使用しています。解凍したDISK 2を起動するとこれらは自動的に組み込まれますが，他のシステムで使用する場合には各自CARD2.FNCおよびCARDDRV.Xを組み込んでください。

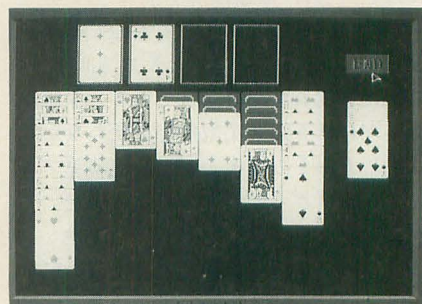



写真2　残念，もはやこれまで

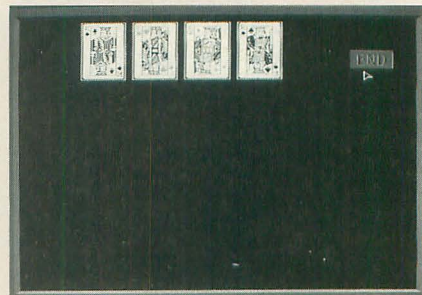
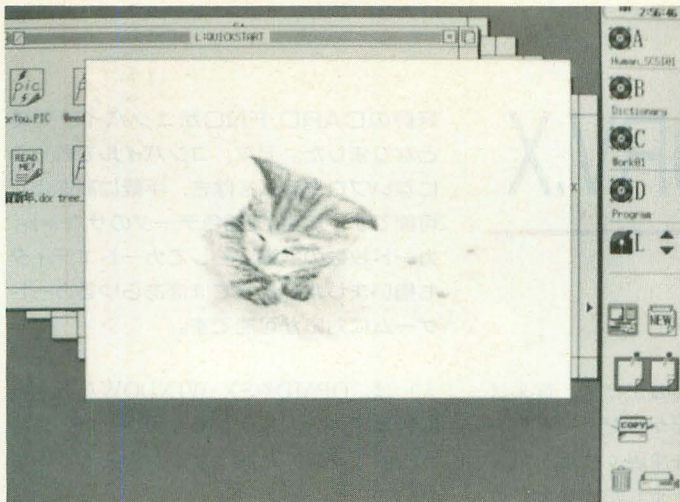
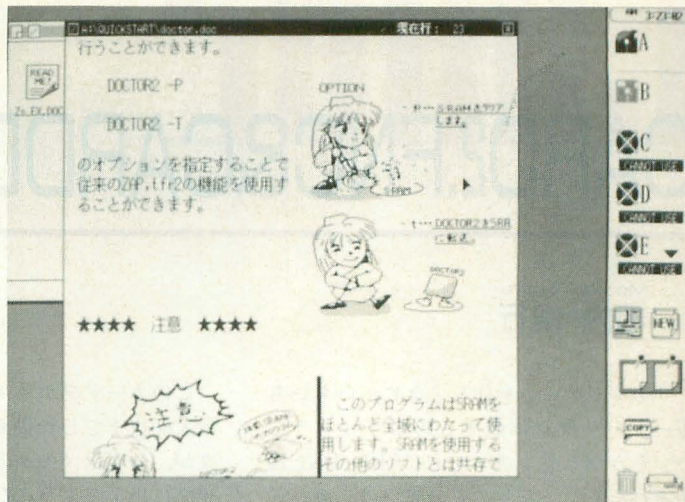



写真3　めでたく完成！

PICファイルの表示例

CUTファイルの表示例

では和音が出てきたときにひどく無駄な処理をしなければならないし，同時発声16音ではMIDIが本来持つ表現力を生かし切れないことになる。MUSICDRV.Xでは1トラックでの和音記述を可能としたことにより合理的なMIDI制御が可能となっている。

●MIDI入出力

MUSICDRV.Xは音楽演奏をするだけでなく，MIDI信号の入出力を制御するための基本的な機能をも備えている。

たとえば，楽器にエクスクルーシブメッセージを送って音色を変更したり，楽器からの信号をバルクダンプすることが簡単にできる。さらに，入力されたMIDIの演奏信号に対し，おおまかな時間情報を加えたものを「録音」することも可能（ただし，直接の再生は不可能）。

●FM音源の拡張機能

FM音源に対してもMIDIライクなベンドやモジュレーションが指定できる。

使用方法

CONFIG.SYSに，

DEVICE = MUSICDRV.X #128 #P64

のような指定を加えると，トラックバッファ容量128Kバイト，サンプリングデータ容量64KバイトのメモリをdoMUSICDRV.Xが組み込まれる。

または，コマンドラインから，

A>MUSICDRV #128 /P64

と指定しても同様の効果がある。

コマンドラインから組み込んだ場合なら，

A>MUSICDRV −R

のようにドライバを解除してメモリを節約することも可能。

OPMDとコンパチで使いたいという場合には組み込み時のオプションを，

MUSICDRV #128 /P400 /Y1 /OPM

のように指定すること。

さて，次にBASICを拡張しなければならない。これには，BASIC.CNFに，

FUNC = MUSIC1

という行をつけ加えるだけでよい。

従来のMUSICDRV.X用のBASIC外部関数はMUSIC2.FNCという名前だった。シャープにはMUSIC2.FNCというファイル名は使わないで，ほしい，といっておいたのだが，C compiler PRO-68K ver.2.0に付属する新しい外部関数もMUSIC2.FNCという名前になっていた。

機能/性能的に見るなら，わざわざMIDI制御用にOPMDRV2.Xを使う人もあまりいないとは思うが，ここは本家の顔を立ててMUSICDRV.X用の外部関数は違う名前に変更する。これ以降，MUSICDRV.X用の外部関数は“MUSIC1.FNC”と呼ぶ。これなら絶対に名前がぶつかることはない。

さて，すでにあるOPMファイルを演奏する場合なら，

A>COPY ABC.OPM OPM1

のように“OPM1”というファイル名にコピーを行うことで演奏が可能だ。

注意点

たとえば，BASICやC言語から音楽演奏をさせていたアプリケーションをコンパイルした場合MUSICDRV.Xではエラーが発生する。OPMDRV.Xを直接呼ぶようなものは実行できない。

使い方にもよるのだが，MUSICDRV.XではOPMDRV.Xよりも多量のメモリを消費する。トラックバッファは大きめにとっておいたほうがよい。

そのほかの詳しい仕様についてはDISK 2の¥MUSICDRV¥MUSIC.DOCを参照してほしい。

PIC & OPM & CUT

DISK 1に含まれるPICファイルはOh!Xのスタッフによる作品だ。VS2.Xとともに搭載されたVSPIC.Xのデモ用サンプルとして収録された。これらはごく当たり前のPICファイルなので，PIC.Rがあれば通常のコマンドラインから表示することもできる。

ForYou.PIC（福原徹）

付録ディスク用に依頼したもの。

Weeds.PIC（福原徹）

当初はあまりに大きかったので，編集部で階調を落としレタッチしてファイルサイズを縮小したもの。残念ながら原作の色合いや立体感がやや損なわれている。

SLEEPCAT.PIC（山田純二）

山田氏の猫3部作のうちいちばん小さかったもの。

TREE−3D.PIC（丹明彦）

トランスピュータを使用したレイトレーシング画像。マッピングなどの小技に頼らないところに迫力がある。作成中のマスターディスクを連続セクタに並べ替えたら思いのほか空き領域が出てきたための穴埋めという説もあるが，やはり作品の質が高かったため収録されたもの。

同様にDISK1に含まれるOPMファイルは，織毛直樹氏のオリジナル曲「イトシキリリモ」と立川正之氏による「小フーガ」だ。著作権関係がほとんど絡まない選曲が特徴的。

また，各ディスクに含まれるカットファイルは，山田純二，福原徹，高橋哲史の各氏が下絵，取り込み，データ変換，ドット修正まで担当してくれたものだ。

CARD2.FNC&CARDDRV.X

好評のCARD.FNCがコンパイラ対応となりました。元々，コンパイルできそうにないプログラムを除き，手軽に高速化が可能です。加えて256色データのサポート，カード枚数の拡張，そしてカードエディタも揃いました。これでほぼあらゆるカードゲームに対応が可能です。

Mounai Toshiyuki
毛内 俊行

「カードゲームを作るときに，いちいちカードデータを作成するのがうっとうしい」そういう思いのもとにCARD.FNCは生まれました。そしてその思いは，カードゲームを作ろうとするすべての人に共通した思いだということが，CARD.FNCを使った数々のゲームプログラムの投稿によって証明されたのではないでしょうか。

そこで，このCARD.FNCをもっと本格的に使ってやろうということで，今回早速CARD.FNCのバージョンアップ版を発表させていただきます。その名もずばり，CARD2.FNCです。

CARD2.FNCの特徴

まずは，CARD2.FNCがCARD.FNCと比べてどのように変わったかを見ていきましょう。

1)　CARD.FNC対応のプログラムがそのまま実行できる

CARD2.FNCはCARD.FNCのアップバージョンなのですから，従来のソフトはまったく改造なしに実行することができます。

2)　カードデータファイルに対応した

CARD.FNCではカードデータを変更するのにはcgetやcsetといった命令を使って，カードを1枚ずつ変更するしか手段はありませんでした。しかし今回，CARD2.FNCに対応させたカードデータファイルの標準形式を設定したことにより，あらかじめメモリ上にロードしたカードデータを自由に表示することができます。

このカードファイルのおかげで，従来は最大60枚までしか扱えなかったカードデータが，今度はメモリの許す限り枚数を気にせずに扱えるようになりました。なお，カードデータファイルの作成には，付属のカードエディタを使うと便利です。

3)　従来の16色モードのほかに256色モードにも対応した

CARD.FNCは，グラフィックモードが16色表示のときでなければカードの表示ができませんでした。しかしカードデータファイルに，あらかじめ使える色数を設定しておくことにより，従来の16色対応のデータ以外に256色対応のデータも扱うことができるようになりました。ただし，65536色モードでのカードの表示は不可能です。

なお，16色対応のカードデータを256色で表示したり，その逆を行ったりすることは可能ですが，表示される絵は当然ぐちゃぐちゃになります。

4)　コンパイラに対応した

コンパイラ用ライブラリの完成により，悲願ともいえるコンパイラへの対応ができるようになりました。もちろん，X-BASICからだけではなくCでも使うことができるようになりました。

こんなふうに従来CARD.FNCで，不満だと思われていた点を，今回は徹底的に改良しました。しかしCARD.FNCでもそうでしたが，カードデータをメモリに置くため，メモリの少ないユーザーの方には使いづらいかもしれません。

特に今回はCARD.FNCのときと異なり，カードデータの管理は専用のデバイスドライバが行っています。そのために，X-BASICを起動する前に必ず，デバイスドライバを組み込んでおかなければならなくなってしまい，少々煩わしくなってしまいました。このドライバはコマンドラインから簡単に組み込むことができます。

とにかく，CARD.FNCとメモリの問題というのは切っても切れない関係です。メモリの少ない人（特にメモリ1Mバイトの人）は，OPMDやSX-WINDOWなどとの同居は諦めて，どうしても使いたい場合はカードドライバを解除してから使ってください。

CARD2.FNCを使う

まず技術的なことはあと回しにして，実際にCARD2.FNCを使うまでの手順を説明しておきましょう。

1)　X-BASICに関数を組み込む

CARD2.FNCは外部関数ですから，最初にX-BASICに組み込んでおく必要があります。この作業は1回やってしまえばあとは必要ないので，早速やってしまいましょう。

最初にX-BASICのディレクトリにCARD2.FNCをコピーします。そしてエディタから，BASIC.CNFなどのコンフィギュレーションファイルを呼び出し，

```
FUNC=CARD2
```

の1行を追加します。いままでCARD.FNCを使っていた人は，

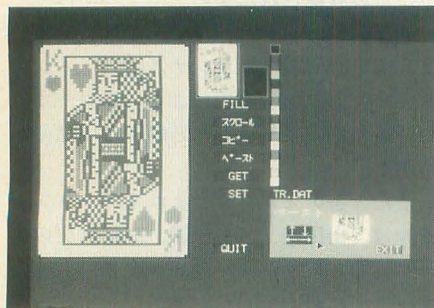
```
FUNC=CARD
```

の1行を変更してください。これで，次回からX-BASICを起動すれば，自動的にCARD2.FNCがX-BASICに組み込まれます。

2)　ドライバを組み込む

CARD2.FNCにはCARDDRV.Xという，専用のデバイスドライバが必要です。このドライバは，カードファイルをメモリにロードして，CARD2.FNCからの指令に対し，表示するカードのデータなどを返しています。

このドライバが組み込まれていないとき，CARD2.FNCはエラーを返すようになっていますので，CARD2.FNC（またはCARD.FNC）対応のプログラムを実行するときには，あらかじめこのドライバを組み込む必要があります。組み込み方は簡単で，コマンドラインから

```
CARDDRV　データファイル名
```

と，実行すれば簡単に組み込むことができます。たとえば，付録ディスクにTR.DATいうCARD.FNCのデータとコンパチなトランプデータファイルがあります。このファイルを組み込むときは，

CARDDRV TR.DAT

と入力してやればいいのです。ただし，カードファイルがカレントドライブにない場合は，ファイル名と一緒にパス名まで指定してください。

また，ドライバを解除したい場合は，

CARDDRV /R

というように，ファイル名の代わりにRオプションを付けます。CARD2.FNCを使わないときは，こうしてドライバを解除すると，カードデータがメモリから開放されて，メモリに余裕ができます。新しいカードデータをロードするときも，このように一度ドライバを開放しなくてはいけません。

さて，これでドライバの組み込み方がわかりました。ただし，X-BASICを起動したり，コンパイルしたプログラムを実行したりするたびにドライバをいちいち起動していたのでは大変ですから，AUTOEXEC.BATなどのバッチファイルに起動させてしまうのが楽でしょう。

ただし，このドライバは特殊な構造をしているために，CONFIG.SYSからDEVICE命令での登録はできないようになっています。組み込みは必ず，バッチファイルかコマンドラインから行うようにしてください。

3) コンパイラを使う

次にコンパイラのライブラリの使い方を説明します。CARDLIB.Aというファイルがそのライブラリで，使い方は，

CC ファイル名 CARDLIB.A

と，コンパイルするときにコンパイルするファイルの後ろに並べて書きます。もし面倒な場合は，BASLIBにアーカイブしてしまいましょう。

さあ，CARD2.FNCの使い方がわかりましたか？ 早速なにか遊んでみましょう！

CARD2.FNCの命令

次にCARD2.FNCを使ってプログラムを書く人のために，その命令を紹介していきましょう。

CARD2.FNCでは従来のCARD.FNCと比べて，X-BASICで扱うことのできる命令の数が増えましたが，もちろん従来の命令もそのままサポートしています（ただし256色対応になったが）。ここでは，その従来の命令も含めてCARD2.FNCで使うことのできる5種類の命令を紹介します。

●c_put(x, y, n)

座標（x，y）で示したグラフィック座標にカード番号nで示したカードパターンを表示します。パラメータはすべてint形式の数値で，返り値はありません。付属のトランプデータを使うときのカード番号と札の関係は，

n＝スート番号×13＋数

で，スート番号はスペード＝0，ハート＝1，ダイヤ＝2，クラブ＝3となっています。

●c_get(n, ca)

カード番号nのカードデータを配列変数caに取り込みます。nはint形式の数，caはchar形式の変数です。返り値はありません。カードデータの大きさは48×96ドットなので，16色モード時で2304バイト，256色モード時で4608バイトが必要です。なお，取り込んだパターンはX-BASICのget, put命令で扱うことができます。

●c_set(n, ca)

カード番号nに，配列変数caに格納されているデータをセットします。つまりc_get命令と逆の動作をするわけです。その他，パラメータの形式や返り値など，c_get命令と同じです。

さて，以上の命令はCARD.FNCでも使われていた命令です。CARD2.FNCではそれ以外に以下の命令が追加されました。

●c_palet( )

表示するカードのパレットを，あらかじめファイルに用意されたパレットテーブルに従って変更する命令です。

ちょっとわかりにくいかもしれませんが，CARD.FNCのトランプデータを例に挙げて見てみましょう。あのデータでは黒の表示部分がカラー番号1（暗い灰色）に設定されています。なぜならば，X68000ではカラー番号0が，黒でなく透明色に設定されているためです。実際にカラー番号を0にしてカードを表示したとすると，バックの色が透けて見えてしまい，情けない結果になってしまうはずです。

99をはじめいままで発表されたゲームのほとんどは，palet(1, 0)などの命令を実行して，カラー番号1を黒くしていますが，色数が最大256色に増えた場合，いちいちパレットを設定するのはやっかいです。あらかじめカードデータファイルには，パレットテーブルを用意してあるので，この命令を実行したらそのテーブルを見てパレットを書き換えるのです。

まだわからない人は，X-BASICを起動して，コマンドラインから，

```
screen 1,1,1,1
c_put(100,0,1)
c_palet( )
```

と実行してみましょう。画面に表示されたスペードのAが，きりっと引き締まるのが見えるはずです。

なお，この命令にはパラメータも返り値もありません。

●c_stat(n)

現在使っているカードファイルのステータスを返します。nはint型の数で0か1を代入します。n＝0のとき，返り値はカードの使っている色数で，必ず16か256が返ります。またn＝1のときの返り値はカードの枚数で，返り値は最大カード番号＋1の値が返ってきます。CARD2.FNCでは使えるカードの色数や枚数が固定ではありませんので，この命令を使ってカードのパラメータを確認できるようにしました。

以上がCARD2.FNCで使える命令です。CARD.FNCの命令に加えて，さらに強力な命令がサポートされました。皆さんもこれらの命令をどんどん使ってください。

デバイスドライバの仕様

さて，ここからしばらくは少し専門的な話になりますので「僕には関係ないや」と思う人はこの先のカードエディタの解説まで読み飛ばしても結構です。ここではマシン語でカード管理ドライバを操作したいという人のために，ドライバの使い方などを説明していこうと思います。

1) デバイスドライバのコマンド

カード管理ドライバCARDDRV.Xのコマンドコードは2バイトの数字から成ります。データをやり取りするときは，読み込み/書き込みモードでドライバをオープンし，2バイトのコマンドをドライバに書き込んだ直後に4バイトのデータをデバイスから読み込みます。ちなみにCARDDRV.Xのデバイス名は“CARDS”です。

また，コマンドの中にはコマンドコードのほかに2バイトのパラメータを一緒にドライバに転送するものがあります。このときは，書き込みバイト数を4バイトとして，コマンドコードーパラメータの順にドライバに書き込んでやります。それでは各コマンドの説明をしていきましょう。

●コマンドコード＄0000
カードデータアドレスの読み込み

指定したカードのパターンが格納されて

いるメモリアドレスを返します。パラメータとしてコマンドコードと一緒にカード番号を2バイト渡します。もし，存在しないカード番号を渡した場合は，デバイスから－1が返されます。

返されたアドレスからのデータは，IOCSコールのPUTGRMを使って表示することができます。

●コマンドコード$0001
パレットテーブルアドレスの読み込み

カードのパレットを定義するパレットテーブルの先頭アドレスを返します。パラメータは存在しません。

●コマンドコード$0002
カードファイル情報の読み込み

カードファイルの情報を読み込みます。返ってきたデータの最初の2バイトはカードの色数，続く2バイトがカードの枚数を示しています。

●コマンドコード$FF00
ドライバを解除するための情報の読み込み

このコマンドはすでにメモリ上にあるCARDDRV.Xを解除するために必要な情報を読み込むためのもので，普段は使いません。このコマンドを実行したときに限り特別に，ドライバからは12バイトの情報が返ってきます。

情報は最初から4バイトごとに，カードデータの先頭アドレス，CARDDRV.X本体の先頭アドレス，直前のデバイスドライバのリンクポインタのアドレスを示しています。CARDDRV.Xを解除するときはこの先頭2つのアドレスをスタックにセットして_MALLOCを実行し，最後の4バイトが示すアドレスに，CARDDRV.Xのリンクポインタの中身をコピーしてあげればいいのです。

以上がドライバのコマンドです。マシン語でカードを表示したい人は使ってみてください。

2)　カードデータファイルの仕様

ドライバが管理するカードデータファイルはそのフォーマットが決まっています。もし，自分で独自にカードデータファイルを作成したりするときは，ファイルのフォーマットをこれから説明する仕様に合わせてください。

カードデータファイルは，ヘッダ，パレットテーブル，カードデータの順に，3つに分けることができます。これからそれらを順に説明していきましょう。

●ヘッダ

ファイルの先頭から16バイトがファイルのヘッダ領域です。ファイルヘッダは先頭から順番に以下のように分けられます。

0～3バイト　：文字列“CARD”
4～5バイト　：カードデータの色数
6～7バイト　：カードデータの枚数
8～15バイト　：未使用領域

最初の4バイトの文字列は，このファイルがCARDDRV.X用のカードデータであることを示すもので，先頭の4バイトは必ず大文字で“CARD”と書かれていなくてはいけません。それからはワード単位で色数，枚数という順に格納されており，それ以降は将来の拡張を考えて未使用ということになっています。

●パレットテーブル

ファイルヘッダの後ろに続くのがパレットテーブルの領域です。ちなみに，CARDDRV.Xはカードデータファイルをメモリ上にロードするとき，このパレットテーブル以降を，そっくりそのままロードします。

パレットテーブルには，各カラーコードのパレットデータが入っており，c_paletなどの命令でパレットを設定するときに使われます。パレットコードは$0000～$FFFFのワード単位で表記されますから256個のパレットを表すためには，テーブルの大きさは全部で256ワード(512バイト)必要になります。

パレットテーブルは先頭からカラーコード0，1……255の順に並んでいます。カードデータが16色のときはカラーコードが16個しか必要ありませんから，テーブルの先頭から32バイト分のパレットデータがセットされ，それ以降には0が入ります。

●カードデータ

パレットテーブルの後ろはカードデータ部です。データ形式はX-BASICのget/put命令やIOCSコールの_PUTGRMで扱うものと同じ形式（16色のときは1バイト＝2ドット，256色のときは1バイト＝1ドットのカラーコードを横方向に羅列したデータ）ですのでカードデータは1枚につき，2304バイト（16色），4608バイト（256色）の大きさが必要となります。

＊　＊　＊

以上がカードデータファイルの内容です。カードエディタを使えば，こんなことは考えなくてもいいのですが，たとえば既存の絵から，カードデータを作成したりしたいときなどは，このファイルフォーマットを参考にして，自分で工夫をしてみてください。ファイルの構造は簡単なので，X-BASICでも手軽に，ファイルの書き換えプログラムが作れることと思います。

カードエディタを使う

さて，今回CARD2.FNCについた特別付録がこのカードエディタです。付属のディスクにはCDEF.Xというファイル名で入っています（シーデフ・エックスと読んでくれれば嬉しいのですが)。

カードエディタは，コマンドラインからCDEFと入力すれば簡単に起動することができます。エディタを実行すると最初に，エディットするカードファイルの設定を行います。

まず，エディットするファイルが新規ファイルかどうか聞いてきますので，新規ファイルならYES，そうでないならNOの位置にマウスカーソルを合わせてマウスの左ボタンをクリックします。このエディタは，カードファイルのファイル名を入力するとき以外，すべての操作はマウスで行うように作られています。

エディットファイルが新規でない場合は，このあとファイル名を入力してすぐにエディットモードになりますので，ここでは詳しい説明はいらないでしょう。さて，新規ファイルを選んだ場合，エディットモードに入る前に新しいカードデータファイルをディスクに作成しなくてはいけませんので，まず新規ファイルを作成する手続きから説明しましょう。

まず，エディットするファイルのファイル名をキーボードから入力します。すると，メッセージウィンドウに

色数　256
枚数　100

と表示されます。このままOKをクリックすると，色数256色，枚数100枚のカードを扱うカードデータファイルが作成されてしまいますので，用途に応じてパラメータを変えてください。変えたい数字の上にマウスカーソルを持ってきてボタンをクリックすれば，その数字を任意の値に変えることができます。

まず色数を変えたい場合，数字の上でマウスボタンをクリックするたびに，256→16→256の順に数が切り替わります。枚数は各桁ごとにボタンをクリックします。

このとき，左ボタンをクリックすると0→1→2……9の順番で数字が切り替わり，右ボタンをクリックすると，9→8→7……0の順番で数字が切り替わります。ただし，100の桁では，数字は0と1が切り替わるだけです。

つまり，このカードエディタで扱うこと

のできるカードの枚数は，最大199枚ということになります。本当はドライバで扱える最大までカードの枚数をサポートしようと思ったのですが（65536枚），メモリの容量を考えると，とてもそんなにたくさんのカードは扱えないのでやめました。それでも256色199枚のカードデータファイルを作成するとそのサイズは，896Kバイトと超キングサイズになってしまいます。フロッピーディスクに納めるにはこれくらいが限界だと思います。

さて，ファイルの設定が終了すると，いよいよカードのエディットに入ります。エディットの方法は，各種グラフィックツールやパターンエディタと基本的には同じなのですぐ使えると思いますが，ひととおりの機能の説明を一応しておきます。

1）色の設定

まず，絵を描くときのペンの色指定の方法ですが，画面の右上にパレットテーブルがありますので，好みの色のところにマウスカーソルを持っていき，マウスの左ボタンをクリックします。

2）絵を描く

画面の左にある格子画面がエディット画面です。マウスカーソルをエディット画面へ持っていき，左クリックをするとエディット画面上に自由に絵を描くことができます。なお，エディット画面上でマウスの右ボタンをクリックすると，マウスカーソルが指している画面上の色をペンの色として設定できます（スポイト機能）。

3）画面の編集機能の説明

CDEF.Xには以上の機能のほかに画面の切り張りやスクロールなど，カードデータの編集用の便利な機能が用意されています。これらの命令は，画面中央にあるメニューを選択することによって使うことができます。以下，それらの命令の説明をします。

なお，これらの機能を使用中，座標などの決定はマウスの左ボタンのクリックで行い，キャンセルやEXITしたい場合はマウスの右ボタンをクリックするか，画面右下に開いている場合はそのウィンドウ内の“キャンセル”や“EXIT”のところで左ボタンをクリックします。

●FILL

指定した長方形の中を指定した色で塗り潰します。この機能を選択すると，マウスカーソルはエディット画面の中しか移動できなくなりますから，画面上で，塗り潰したい長方形のかどを，左上－右下の順で指定してください。

●スクロール

エディット画面を指定方向に1ドットずつスクロールさせます。機能メニューからスクロールを選択すると，画面右下にメッセージウィンドウが開き，上下左右4方向を向いた矢印が表示されるので，マウスカーソルをスクロールさせたい方向の矢印の所へ運んでクリックしてください。

●コピー/ペースト

エディット画面上で指定した長方形の中に描かれたパターンを写し取り，別の場所に張り付けます。なお張り付けるときに，写し取ったパターンを上下，左右に反転することができます。

まず，パターンを写し取るには機能メニューのコピーを選択します。写し取る領域の指定は，FILLのときと同じ要領で，左上－右下の順に長方形の指定を行ってください。指定された長方形の内容が，エディタのカットバッファにセットされます。

写し取ったパターンの張り付けはペーストを使います。機能メニューからペーストを選択すると，画面の右下にメッセージウィンドウが開き，写し取ったパターンが表示され，その横に“↑↓”と“←→”の2つの矢印が表示されます。“↑↓”の上にマウスカーソルを持っていき左クリックをすると，ウィンドウ内のパターンが上下反転を“←→”でクリックをすると，左右反転をするのが見えます。

張り付けるときはマウスカーソルをエディット画面の中に持っていき，任意の場所で左クリックをします。ただしマウスカーソルがエディット画面の中にあっても，張り付けるパターン全体がエディット画面に入りきらないときは，クリックをしても張り付けは行われません。張り付けが可能なときは，エディット画面上に張り付ける領域の枠を赤く表示します。

またパターンの張り付けは，連続して何カ所にも張り付けることができるように，1回張り付けてもエディットモードには戻らないようになっています。エディットモードに戻りたいときは，右クリックをするか，メッセージウィンドウの“EXIT”のところで左ボタンをクリックしてください。

●パターンのGET/SET

すでにカードデータファイルにあるカードパターンをエディット画面上に取り出したり，エディット画面上で描いたカードパターンをカードデータファイルにセットしたりします。

カードデータファイルからエディット画面上にパターンを取り出すにはGETを使います。選択すると，画面いっぱいにファイルの中に格納されているカードが表示されるので，任意のカードの上にマウスカーソルを持っていき左クリックをします。

ただし1画面に表示できるカードの枚数は40枚までです。カードの枚数がそれ以上のときで，41枚目以降のカードを選択したいときは，画面の右下の“NEXT”をクリックしてください。

エディットしたカードパターンをファイルにセットしたいときはSETを使います。まず，GETのときと同じように画面いっぱいにカードが表示されますので，任意のカードデータのところへマウスカーソルを持っていき左クリックをすると，エディットしたカードがファイルにセットされます。

ただしプログラムの都合上，256色モードでデータのセットを行うと，ファイルをセットするのに5～6秒の時間がかかります。16色のときは気にするほど時間はかかりません。

4）エディットするファイルの変更

パレットテーブルの下に，現在エディット中のファイルの名前が表示されていますが，そのファイル名の上でマウスをクリックするとファイルの変更が可能です。設定方法はこのエディタを起動したときとまったく同じで，新規ファイルかどうかの確認やファイル名の入力などを行います。

5）終了

エディットが終わったら，エディタを終了しHuman68kへ戻らなくてはいけません。終了させる方法は，機能メニューの下にある“QUIT”を選択します。また，ファイル名の入力時にファイル名の代わりにCTRL-C(BREAK)を入力しても，エディタを終了します。

以上がカードエディタの使い方です。私がCARD.FNCのデータを作る際も，これとほとんど同じ仕様のカードエディタを独自に作って使っていたので，皆さんも根気さえあれば自分独自のオリジナルカードが作れると思います。どんどん活用してください。

プログラムリストについて

参考までに，ドライバのソースリストを載せておきます。が，綺麗なプログラムではないので，あまり参考にならないと思います。特にカードエディタに至っては，締め切りに追われながら無計画に作ったために，プログラムそのものはえらく醜いものになってしまいました。

それでもこのエディタの場合，過去に作ったMZ-2500のプリンタ用外字エディタ（Oh!X1988年11月号）やCARD.FNCのカードパターンを作ったカードエディタなどの経験が生きていたので，プログラムは醜くても結構まともなでき栄えになりました。

しかし，いままでまったく手をつけたことのなかったデバイスドライバなどは，参考になるようなプログラムに仕上がったのかどうか，完成したいまでもあまり自信がありません。しかも，都合によりCONFIG.SYSからの登録をできなくするために，デバイスヘッダをデバイスの先頭に持ってきていませんので，醜いプログラムが余計に複雑になってしまった感があります。

ただ，デバイスドライバなどのソースには，1行ずつちゃんとコメントを入れていますので（これは実は自分のためだったのですが）これらのコメントと命令を見比べれば，見づらいリストも少しは見やすくなると思います。

なお，これらのプログラムの作成にあたり，カードエディタの作成にはX-BASICとGCCのVer.1.36.01を使いました。また，その他のプログラムはすべて，XCに付属のアセンブラ，AS.Xを使っています。

最後に

いやー，今回の仕事は実際とてもヘビーでした。だいたい最初に締め切りまでの猶予期間が1カ月はあったはずなのに，半月たってもドライバが完成していなかったり（いったん完成はしたのだけど，ボツになったのだった）コンパイラのライブラリに原因不明のバグが発生したりと，アクシデントだらけ。最初のうちは「1カ月でできるね？」という編集部の問いかけに「大丈夫です」とか「まかせてください」などと気前のいい返事をしていたのが，だんだん「頑張ります」とか「やってみます」などという返事が多くなってきて，ついには「本当に完成するのだろうか……」と途方に暮れてしまっていました。

とにかくシステムプログラムというのは作っていて思いっ切りストレスが溜まります。私はこの期間，ストレスが溜まるとYet Another Columnをやりストレスを解消していました。おかしなもので，ストレスが溜まっているときにこのゲームをやるとスコアが飛躍的に伸びます。この期間に記録したスコアはトップの43480点を最高に4万3千点台を立て続けに3つも記録しました。いま，挑戦してもこのスコアの足元にも及ばないのですから不思議なものです。

話が横にそれましたが，そんなこんなの苦労のうちに，無事にCARD2.FNCも完成し，こうして無事皆さんに配布することができました。これもすべて，私が行き詰まるたびに数々のアドバイスや励ましの言葉をいただいた編集部やそのスタッフの方々のおかげです。ここに改めてお礼申し上げます。特に，本誌連載のX68000マシン語プログラミングは，デバイスドライバを作ったことのない私にとってプログラマーズマニュアルと並ぶ数少ない資料として活用しました。村田敏幸氏に深く感謝いたします。

その他，開発期間中に数々のわがままを聞いてくださった人たち，特に日本大学コントラクトブリッジクラブの方々に深く感謝します。それでは皆さん，これからもCARD2.FNCをよろしくお願いします。

リスト1

```
==================  CARDDRV.S  ==================
  1:            .include        DOSCALL.MAC
  2:            .include        IOCSCALL.MAC
  3:
  4: Human:             equ     $6800   *Human68kの先頭アドレス
  5: NULATR:            equ     $8024   *NULデバイスの属性
  6:
  7: pstart:
  8:            adda.l  #1,a2
  9:            move.l  a0,memptr       *メモリポインタ待避
 10:            move.l  a1,endadr       *プロセス終了アドレス待避
 11:            move.l  a2,comline      *コマンドラインポインタ待避
 12:
 13:            lea.l   16(a0),a0       *a0=メモリブロックポインタ
 14:            suba.l  a0,a1           *a1=プロセスのバイト数
 15:            move.l  a1,-(sp)        *それぞれスタックに
 16:            move.l  a0,-(sp)        *セットして
 17:            DOS     _SETBLOCK       *余分なメモリを開放
 18:            addq.l  #8,sp
 19:
 20:            pea.l   TitleName       *一丁前にタイトルなど
 21:            DOS     _PRINT          *表示するのだ
 22:            addq.l  #4,sp
 23:
 24:            bsr     drvchk          *ドライバのチェック
 25:            tst.l   d0              *既にCARD.Xが組み込まれていたら
 26:            bmi     already         *その時の処理へ
 27:
 28:            clr.w   -(sp)           *読み込みモードで
 29:            move.l  comline,-(sp)   *指定ファイルを
 30:            DOS     _OPEN           *オープン
 31:            addq.l  #6,sp
 32:            tst.l   d0              *オープンできなければ
 33:            bmi     opnerr          *エラー
 34:            move.w  d0,handle       *ファイルハンドル格納
 35:
 36:            move.l  #16,-(sp)       *size=8bytes
 37:            pea.l   headbf          *ファイルヘッドポインタへ
 38:            move.w  handle,-(sp)    *指定ファイルを
 39:            DOS     _READ           *読み込む
 40:            lea.l   10(sp),sp
 41:
 42:            lea.l   headbf,a0       *a0=ファイルヘッドポインタ
 43:            move.l  #'CARD',d0      *ファイルの先頭4バイトが
 44:            cmp.l   (a0),d0         *"CARD"か?
 45:            bne     nofile          *違ったらエラーいこっちゃ
 46:
 47:            moveq.l #0,d0
 48:            moveq.l #0,d1
 49:            move.w  4(a0),d0        *d0=データの色数
 50:            move.w  6(a0),d1        *d1=カードの枚数
 51:            mulu.w  #2304,d1        *d1=カードデータの大きさ
 52:            cmp.w   #16,d0          *カードの色数が
 53:            beq     mlc             *16色じゃなかったら
 54:            add.l   d1,d1           *d1を更に2倍にする
 55: mlc:
 56:            add.l   #512,d1         *パレットテーブルの大きさを加え
 57:            move.l  d1,size         *データサイズを格納
 58:            move.l  size,-(sp)      *それだけの大きさの
 59:            DOS     _MALLOC         *メモリを確保
 60:            addq.l  #4,sp
 61:            tst.l   d0              *メモリを確保できなければ
 62:            bmi     memerr          *エラー
 63:            move.l  d0,dntptr       *データポインタを格納
 64:
 65:            move.l  size,-(sp)      *データサイズと
 66:            move.l  datptr,-(sp)    *読み込む先のポインタを指定して
 67:            move.w  handle,-(sp)    *指定ファイルから
 68:            DOS     _READ           *カードデータを読み込む
 69:            lea.l   10(sp),sp
 70:
 71:            move.w  handle,-(sp)    *そして指定ファイルを
 72:            DOS     _CLOSE          *クローズする
 73:            addq.l  #2,sp
 74:
 75:            pea.l   compms
 76:            DOS     _PRINT
 77:            addq.l  #4,sp
 78:
 79:            clr.l   -(sp)           *最後にプロセスを常駐終了
 80:            move.l  #pend,d0        *するのであった
 81:            sub.l   #pstart,d0      *
 82:            move.l  d0,-(sp)        *
 83:            DOS     _KEEPPR         *バイビー
 84: *
 85: already:
 86:            movea.l comline,a0      *Rオプションしかないので
 87:            move.b  (a0)+,d0        *とっても手抜きしています。
 88:            cmp.b   #'/',d0         *参考にしないでね
 89:            beq     al2             *
 90:            cmp.b   #'-',d0         *
 91:            bne     iloper          *
 92: al2
 93:            move.b  (a0),d0         *
 94:            cmp.b   #'R',d0         *
 95:            beq     unlock          *
 96:            cmp.b   #'r',d0         *
 97:            beq     unlock          *
 98: iloper:
 99:            pea.l   ilopms
100:            DOS     _PRINT
101:            addq.l  #4,sp
102:            DOS     _EXIT
103: unlock:
104:            pea.l   unlockms
105:            DOS     _PRINT
106:            addq.l  #4,sp
107:
108:            move.w  #2,-(sp)        *読み込み/書き込みモードで
109:            pea.l   DEVNAM          *ドライバを
110:            DOS     _OPEN           *オープン
111:            addq.l  #6,sp
112:            move.w  d0,DEVHDL       *ファイルハンドル格納
113:
114:            move.l  #2,-(sp)        *size=2bytes
115:            pea.l   COM_FF00        *メモリ解除アドレスを聞くため
116:            move.w  DEVHDL,-(sp)    *ドライバにコマンドを
117:            DOS     _WRITE          *転送
118:            lea.l   10(sp),sp
119:
120:            move.l  #12,-(sp)       *size=12bytes
121:            pea.l   retdat          *バッファに
122:            move.w  DEVHDL,-(sp)    *デバイスからの返事を
123:            DOS     _READ           *読み込む
124:            lea.l   10(sp),sp
125:
126:            move.w  DEVHDL,-(sp)    *ドライバを
127:            DOS     _CLOSE          *クローズ
128:            addq.l  #2,sp
129:
130:            move.l  #0,a1
131:            IOCS    _B_SUPER        *スーパーバイザモードに移行
132:            move.l  d0,sspbf
133:
134:            lea.l   retdat,a0       *直前のデバイスの
135:            movea.l 8(a0),a0        *リンクポインタを書き変える
136:            move.l  l_point,(a0)    *
137:
```

```
138:         movea.l sspbf,a1
139:         IOCS    _B_SUPER            *ユーザーモードに移行
140:
141:         lea.l   retdat,a0
142:         move.l  (a0),-(sp)          *カードデータと
143:         move.l  4(a0),-(sp)         *デバイスの
144:         DOS     _MFREE              *メモリを解除する
145:         addq.l  #4,sp
146:         DOS     _MFREE
147:         addq.l  #4,sp
148:
149:         DOS     _EXIT               *非常中終了
150:
151: drvchk:
152:         movea.l #0,a1
153:         IOCS    _B_SUPER            *スーパーバイザモードへ移行
154:         move.l  d0,sspbf            *sspを保存
155:
156:         lea.l   Human,a0            *a0=Human68kの先頭アドレス
157:         move.w  #'NU',d0            *d0="NU"           NULデバイスの
158:         move.l  #'L   ',d1          *d1="L   "         デバイス名を
159:         move.w  #'  ',d2            *d2="  "           d0,d1,d2に格納
160: dc1:
161:         cmp.w   (a0)+,d0            *デバイス名の最初の2バイトを比較
162:         bne     dc1                 *違ったら繰り返す
163:         cmp.l   (a0),d1             *次の4バイトを比較
164:         bne     dc1                 *違ったら繰り返す
165:         adda.l  #4,a0               *a0=a0+4
166:         cmp.w   (a0),d2             *最後の2バイトを比較
167:         bne     dc1                 *違ったら繰り返す
168:         cmp.w   #NULATR,-16(a0)     *デバイスの属性を比較
169:         bne     dc1                 *違ったら繰り返す
170:
171:         suba.l  #20,a0              *a0=NULデバイスの先頭アドレス
172:         move.l  #'CARD',d0          *d0=カードデバイス名の前4バイト
173:         move.l  #'S   ',d1          *d1=カードデバイス名の後4バイト
174: dc2:
175:         move.l  14(a0),d2           *デバイス名の前4文字を
176:         cmp.l   d2,d0               *比較
177:         bne     dc3                 *一致したら
178:         move.l  18(a0),d2           *続く後4文字も
179:         cmp.l   d2,d1               *比較
180:         beq     dc4                 *一致すれば処理を移す
181: dc3:
182:         cmp.l   #-1,(a0)            *リンクポインタ=-1?
183:         beq     dc5                 *もしそうなら処理を移す
184:         movea.l (a0),a0             *a0=次のデバイスの先頭
185:         bra     dc2                 *繰り返し
186: dc4:
187:         movea.l sspbf,a1
188:         IOCS    _B_SUPER            *ユーザーモードへ移行
189:         move.l  #-1,d0              *見つからなかったと知らせに
190:         rts                         *帰る
191: dc5:
192:         move.l  a0,endlink          *前のデバイスのポインタを格納
193:         move.l  #l_point,(a0)       *リンクポインタを書き変える
194:         movea.l sspbf,a1
195:         IOCS    _B_SUPER            *ユーザーモードへ移行
196:         moveq.l #0,d0               *見つかったと知らせに
197:         rts                         *帰る
198: opnerr:
199:         pea.l   operms
200:         bra     erprt
201: nofile:
202:         pea.l   noflms
203:         bra     erprt
204: memerr:
205:         pea.l   meerms
206: erprt:
207:         DOS     _PRINT
208:         addq.l  #4,sp
209:
210:         move.l  #0,a1               *
211:         IOCS    _B_SUPER            *スーパーモードへ移行
212:         movea.l endlink,a0          *書き変えてしまったリンク
213:         move.l  #-1,(a0)            *ポインタを元に戻す
214:         move.l  d0,a1
215:         IOCS    _B_SUPER            *ユーザーモードへ移行
216:
217:         move.w  handle,-(sp)        *開いたかも知れないファイルを
218:         DOS     _CLOSE              *クローズして
219:         addq.l  #2,sp
220:
221:         DOS     _EXIT               *非常中終了
222: *
223: *  デバイスドライバ部
224: *
225: COMCODE         equ     2
226: ERROR_L         equ     3
227: ERROR_H         equ     4
228: DEV_END         equ     14
229: BFADR           equ     14
230: DATLEN          equ     18
231: *
232: *  デバイスヘッダ
233: *
234: l_point:
235:         .dc.l   -1                  *リンクポインタ
236:         .dc.w   $8020               *デバイス属性(CHR / RAW)
237:         .dc.l   str_ent             *ストラテジルーチンエントリ
238:         .dc.l   int_ent             *割り込みルーチンエントリ
239:         .dc.b   'CARDS   '          *デバイス名
240: *
241: *  ストラテジルーチン
242: *
243: str_ent:
244:         move.l  a5,r_head           *リクエストヘッダを待避して
245:         rts                         *さようなら
246: *
247: *  割り込みルーチン
248: *
249: int_ent:
250:         movem.l d0-d7/a0-a6,-(sp)
251:         movea.l r_head,a5
252:         moveq.l #0,d0
253:         move.b  COMCODE(a5),d0
254:         add.w   d0,d0
255:         add.w   d0,d0
256:         lea.l   j_table,a4
257:         adda.l  d0,a4
258:         movea.l (a4),a4
259:         jsr     (a4)
260:         move.b  d0,ERROR_L(a5)
261:         lsr.w   #8,d0
262:         move.b  d0,ERROR_H(a5)
263:         movem.l (sp)+,d0-d7/a0-a6
264:         rts
265: *
266: *  コマンドのジャンプテーブル
267: *
268: j_table:
269:         .dc.l   notcom              *初期化
270:         .dc.l   notcom              *エラー
271:         .dc.l   notcom              *未使用
272:         .dc.l   notcom              *IOCTRL入力
273:         .dc.l   input               *入力
274:         .dc.l   okcom               *先読み入力
275:         .dc.l   okcom               *入力ステータスチェック
276:         .dc.l   okcom               *バッファクリア
277:         .dc.l   output              *出力(verify off)
278:         .dc.l   voutput             *出力(verify on)
279:         .dc.l   okcom               *出力ステータスチェック
280:         .dc.l   notcom              *未使用
281:         .dc.l   notcom              *IOCTRL出力
282: *
283: *  各コマンドの処理
284: *
285: notcom:
286:         move.w  #$5030,d0           *エラーコードをセットして
287:         rts                         *さよーなら
288: input:
289:         move.l  DATLEN(a5),d0       *d0=バイト数
290:         movea.l BFADR(a5),a0        *a0=リターンバッファ
291:         lea.l   retdat,a1           *a1=データバッファ
292: inpl:
293:         move.b  (a1)+,(a0)+         *データ転送
294:         dbra    d0,inpl
295: okcom:
296:         moveq.l #0,d0               *無事終了という印を持って
297:         rts                         *ばいばい
298:
299: output:
300: voutput:
301:         movea.l BFADR(a5),a0
302:         move.w  (a0)+,d0            *d0=コマンドコード
303:
304:         cmp.w   #0,d0               *コマンドコード=0000?
305:         beq     askadr              *データアドレスGETへ
306:         cmp.w   #1,d0               *コマンドコード=0001?
307:         beq     askpal              *パレットアドレスGETへ
308:         cmp.w   #2,d0               *コマンドコード=0002?
309:         beq     askstat             *ステータスGETへ
310:         cmp.w   #$FF00,d0           *コマンドコード=FF00?
311:         beq     sysadr              *解除アドレスGETへ
312:         bra     okcom
313: askadr:
314:         lea.l   headbf,a1
315:         move.w  6(a1),d1            *d1=カードの枚数
316:         move.w  (a0),d0             *d0=カード番号
317:         cmp.w   d0,d1               *もし、d1<d0なら
318:         bls     asa2                *エラー
319:         mulu.w  #2304,d0
320:         move.w  4(a1),d1
321:         cmp.w   #16,d1
322:         beq     asa1
323:         add.l   d0,d0
324: asa1:
325:         add.l   datptr,d0
326:         add.l   #512,d0
327:         move.l  d0,retdat
328:         bra     okcom
329: asa2:
330:         move.l  #-1,retdat
331:         bra     okcom
332: askpal:
333:         move.l  datptr,retdat
334:         bra     okcom
335: askstat:
336:         lea.l   headbf,a1
337:         move.l  4(a1),retdat
338:         bra     okcom
339: sysadr:
340:         lea.l   retdat,a1
341:         move.l  datptr,(a1)+
342:         move.l  memptr,d0
343:         add.l   #16,d0
344:         move.l  d0,(a1)+
345:         move.l  endlink,(a1)
346:         bra     okcom
347:
348: memptr:                             *メモリ管理ポインタアドレス
349:         .dc.l   0
350: endadr:                             *プロセス終了アドレス+1
351:         .dc.l   0
352: comline:                            *コマンドラインポインタアドレス
353:         .dc.l   0
354: handle:                             *ファイルハンドル
355:         .dc.w   0
356: headbf:                             *ファイルヘッド格納領域
357:         .dc.l   0,0,0,0
358: size:                               *確保したメモリサイズ
359:         .dc.l   0
360: datptr:                             *データの先頭アドレス
361:         .dc.l   0
362: sspbf:                              *sspスタックの待避領域
363:         .dc.l   0
364: r_head:                             *リクエストヘッダ待避領域
365:         .dc.l   0
366: retdat:                             *デバイスからのデータのバッファ
367:         .dc.l   0,0,0
368: endlink:                            *手前のデバイスのリンクポインタ
369:         .dc.l   0
370: DEVNAM:                             *ドライバのファイルネーム
371:         .dc.b   'CARDS',0
372: DEVHDL:                             *ドライバのファイルハンドル
373:         .dc.w   0
374: COM_0:                              *カードアドレスを聞くコマンド
375:         .dc.w   0
376: P_COM0:                             *上記コマンドのパラメータ
377:         .dc.w   0
378: COM_1:                              *パレットテーブルアドレスを聞く
379:         .dc.w   1
380: COM_2:                              *カードデータの色数/枚数を聞く
381:         .dc.w   2
382: COM_FF00:                           *システム関係のコマンド
383:         .dc.w   $FF00
384: TitleName:
385:         .dc.b   13,10
386:         .dc.b   'カード管理 DRIVER for X68000 version 1.00'
387:         .dc.b   13,10
388:         .dc.b   'Presented by Oh!X / Toshiyuki Mounai / 1990.OCT'
389:         .dc.b   13,10,0
390: operms:
391:         .dc.b   '指定したファイルをオープンできませんでした',13,10,0
392: noflms:
393:         .dc.b   'このファイルはカードデータではありません',13,10,0
394: meerms:
395:         .dc.b   'メモリを確保できませんでした',13,10,0
396: unlockms:
397:         .dc.b   'ドライバを解除しました',13,10,0
398: ilopms:
399:         .dc.b   'オプションの指定に誤りがあります',13,10,0
400: compms:
401:         .dc.b   'カードデータを無事登録しました',13,10,0
402: pend:
```

Z'sSTAFF支援ツール"Z's-EX"

グラフィックツールの最高峰としてそびえるZ'sSTAFF　PRO-68K。でも，"PRO"ならばもっと強力な機能がほしいもの。そこで日頃もの足りなく思っていた機能のいくつかを拡張/強化してみましょう。残念ながら，完全に使うには4Mバイト（?)のメモリが必要です。

Tan Akihiko
丹 明彦

第1章　今回のプログラムは

今回のプログラム「Z's-EX」は，Z'sSTAFF PRO-68Kの機能を拡張し，ユーザーが絵を描く作業をサポートするツールです。

拡張といっても，Z'sSTAFFそのものを改造するわけではありません。Z's-EXは，通常はメモリ空間のどこかで待機しています。そのあいだは，ふつうとまったく同じように描画作業を行うことができます。ユーザーは，Sキーを押すことで，Z's-EXを呼び出すことができます。Z's-EXを呼び出しているあいだは，通常のZ'sSTAFFの機能を使うことはできません。Z's-EXでの作業を終えて，Z's-EXを抜けて初めてZ'sSTAFF本来の作業を続けることができます。

Z's-EXは，Z'sSTAFFとはまったく独立に動作してユーザーの作業を助けるアプリケーションです。

Z's-EXは，呼び出し元のプログラムがZ'sSTAFFかどうかを一切チェックしていないので，Z's-EXはZ'sSTAFFのバージョンに関係なく使用できるはずです。原理的には，Z'sSTAFFでなくてもいいのですが，それはちょっと事情があってできません。

このプログラムのアイデア，つまりZ'sSTAFFと同居して独立に動作するアプリケーションという考え方は，電脳倶楽部に掲載された「PICFILER」というプログラムから拝借しました（実はソースリストからも少々失敬しています)。これは，いまやX68000の標準画像フォーマットとして定着した感のあるPICファイルを，Z'sSTAFFからロードおよびセーブできるようにしたツールです。これを使うことで作業効率は飛躍的に上がりました。正直いって，Z'sSTAFF標準のZIMファイルは作業をするうえでネックになっていたのですから。

メインメニュー

Z's-EXは，このPICFILERを下敷きにして，あると便利そうな機能をいくつか付け加えたものです。編集部で検討の結果，今回のバージョンでは以下の機能を持たせることにしました。

・PICFILER……これは当然です。

・裏画面とその切り替え……Z's-EXは，ちょっとした作業場として，また後述する画面合成のソースとして，裏画面を1枚持っています。したがって，システムは，Z'sSTAFFで扱うメイン画面(G-RAM)と裏画面の2枚を持つことになります（細かいことをいえばまだほかにも画面はあるのですが，それはひとまず置いておきます)。両者はいつでも切り替えられます。切り替えれば，裏は表に，表は裏になるのですから，当然，どちらの画面に対してもZ'sSTAFFの機能は使えることになります。

・3次元マッピング……裏画面を1枚の長方形と考えて，画面の中に広がる空間に飛ばしたと思ってください。その板は回転も移動もできます。場所が決まったら，そこに張り付けることができます。すると，裏画面の絵がパースのかかった状態で出現するのです。

ちなみにこれは，Z'sSTAFFの変形とは違います。あちらは文字どおりの変形で，用途がまったく別なのです。

なお，移動も回転もしなければ，それは裏画面からのコピーと同じになります。マスキングを上手に使えば，画面合成にも応用できます。なお，コピー用のときだけは高速に処理されます。

・マスキングのペイント……Z'sSTAFFはマスキング機能を持っています。あれはなかなか強力なのですが，どうも使い勝手が悪い。たとえば人物の絵を画面の真ん中に描きます。その人物の周りだけをマスクしたいと思ったとします。当然ペイントしたいと思います。できません。いやできないというのは正しくありません。正確には，全画面がべったりとマスクで塗りつぶされてしまいます。しかたがないので，ペンを使って人物の輪郭に沿って丁寧にマスクし，しかるのちに周りをペイントします。ああ文化的。

マスクは作業画面とは独立に描画する，という考えで作られています。これはこれで間違ってはいません。しかし不便に感じる局面が多く発生するのも事実です。そこで，単色領域を塗りつぶしてマスクに変えるペイントルーチンを用意しました。マウスでちょいと指定するだけで一瞬にして人物の周りだけをマスキングできるのです。これは画面合成の際，大いに威力を発揮することでしょう。

・特殊効果……あったら面白そうな画像処理機能を4つほどつけてみました。いずれも，マウスで囲んだ領域の内側に処理を施すようになっています。

1)　白黒化処理：読んで字のとおりです。

2)　ランダムフラクタル処理：本誌1990年9月号の特集で紹介したものを少し高速にしたものです。自然物の疑似的な表現や，レイトレーシングのためのマッピングデータ生成ツールとしてお使いいただけることでしょう。

3)　フレア処理：明るい部分を光源に見立て，その周りにぼんやりと光を滲ませる処理です。光源部（?）はマスクで指定します。

4)　微分処理：名前は難しそうです。しかしできあがりは単純明快。石でできたレリーフのような効果が出せます。取り込み画像に対してかけると，なかなか味のある絵になります。

第2章　インストールのしかた

さて，Z's-EXを使うためには，お使いのシステムにインストールしなくてはなりません。

インストール作業は簡単です。Zs_EX.Xとrf.datおよびmapicon.datをZ'sSTAFFと同じディレクトリに置くだけです。ただ，ディスク容量の関係からフロッピーで使う場合にはそれなりの工夫が必要なこと，そして，rf.datは付属の自動生成プログラム(rfbuild.x)を使って作らなくてはならないこと，この2点には注意しておいてください。

(メインメモリ1Mバイトの方へ)

Z's-EXは，当然のことながらZ'sSTAFF単独で使っているときよりもメモリを消費します。そこで非常に残念なお知らせがあります。Z's-EXシステムは，メインメモリ1MバイトのX68000では動作しません。

もうこれはどうしようもありません。裏画面に512Kバイトも取っているのですから，まあ当然の結果とはいえますが。

(メインメモリ2Mバイトの方へ)

メインメモリ2Mバイトあれば，Z's-EXシステムは動作します。ただ，依然として厳しい状況には変わりありません。アンドゥも使えないようです。裏画面を有効に使って対処してください。

(フロッピーディスクでお使いの方へ)

Zs_EX.Xは起動時に2つのデータファイルを読み込みます。rf.datとmapicon.datです。そのあと，STAFF68K.X(Z'sSTAFFの実行ファイル)をチャイルドプロセスとして呼び出します。これらのファイルがないとエラーになってしまいます。

以上の理由から，ソースリストを書き換えて再コンパイルしない限り，STAFF68K.XはZs_EX.Xと同じディレクトリに置いておかなくてはなりません。また，rf.datとmapicon.datも同じディレクトリに置いておく必要があります。

Z'sSTAFFのシステムディスクの容量は，ただでさえかなり苦しい状況にあります。そこへ持ってきて，Z's-EXのけっこう大きなファイルをいくつも転がさなくてはなりません。したがって，これまたまことに残念な話ですが，Z's-EXとZ'sSTAFFが両方入ったシステムは，1枚のフロッピーディスクには入りません。

ハードディスクユーザーの人には，まったく問題はないのですが，フロッピーディスクだと，まずインストールから悩まなくてはなりません。

(というわけで解決編です)

現在のX68000ユーザーの多くは，「メインメモリ2Mバイト，ハードディスクなし」というシステム構成を取っていると思います。したがって，上で述べたようなトラブルに巻き込まれる可能性は非常に大きいものといわなくてはなりません。

この問題は，ディスクを2枚組にし，なおかつ最低限のデバイスドライバを組み込み，使える記憶領域を最大限にすることで解決することにします。

以下の説明は，「メインメモリ2Mバイト，ハードディスクなし」のシステムに対するインストール方法についてです。メモリがもっとあるとか，ハードディスクをつけているとか，そういう人は適宜変更してください。

2枚組になったディスクのうち，AドライブにはHuman68kのシステムやASK68Kの辞書，BドライブにはZ'sSTAFFおよびZ's-EXの実行ファイルを入れることにします。図1をご覧ください。両ドライブに入れるべきファイルの一覧と，CON

2MバイトRAM，ハードディスクなしのユーザーのための
システムディスク構築例

ディスクは2枚用意します。
Z's-EX 関係のファイル以外のファイルは，Human68k システムディスクおよび Z'sSTAFF PRO-68K システムディスクからコピーしてください。

［ドライブ0のファイル］

```
¥
├─HUMAN.SYS          以下は Human68k のファイルです。
├─CONFIG.SYS
├─COMMAND.X
├─AUTOEXEC.BAT
├─SYS
│  ├─FLOAT2+.X       ←もちろんFLOAT2.XやFLOAT3.Xでも構いません。
│  └─ASK68K.SYS
└─DIC
   ├─X68K_M.DIC
   └─X68K_S.DIC
```

［ドライブ1のファイル］

```
¥
├─STAFF68K.X         以下は Z'sSTAFF PRO-68K のファイルです。
├─STAFF68K.SYS
├─CHR08.DAT
├─ICON.DAT
├─ICON3.DAT
├─ICON2.DAT
├─ICON4.DAT
├─MPALET.PAL
├─WPALET.WPL
├─PCGWDW.PDT
├─PEN.PN3
├─TILE.TIL
├─TONE.TON
├─STAFF.TMP
│                    以下は Z's-EXのファイルです。
├─Zs_EX.X            ← 付録ディスクに入っています。
├─mapicon.dat        ← 付録ディスクに入っています。
└─rf.dat             ← 付録ディスクには入っていません。
                       rfbuild.x を用いて作ってください。
```

［A:¥CONFIG.SYSの内容］

```
FILES      =20
BUFFERS    =20 1024
*DEVICE    =A:¥SYS¥ASK68K.SYS A:¥DIC¥X68K_M.DIC A:¥DIC¥X68K_S.DIC
DEVICE     =A:¥SYS¥FLOAT2+.X
```

［A:¥AUTOEXEC.BATの内容］

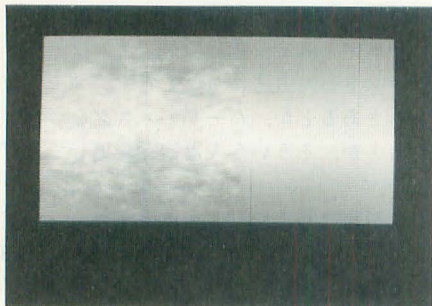
```
ECHO OFF
B:
Zs_EX
A:
```

ランダムフラクタル

FIG. SYSおよびAUTOEXEC. BATの内容が書いてあります。これにしたがってディスクを作り上げてください。メモリを食うようなものはできるだけCONFIG. SYSに書かないよう心掛けてください。それだけZ'sSTAFFで使えるメモリが減りますので，いろいろと制限が出てくるのです。実のところ，ASKもあまりすすめられないのです。もちろん，4Mバイトほどメモリを積んでいれば，なんの支障もありません。

この章の最初で申し上げたとおり，ランダムフラクタル用のデータファイル（rf. dat)は自分で作らなくてはなりません。付録ディスクを解凍しても，ついてこないのです。実はrf. datは，なぜかZ's-EXシステムのなかではもっとも大きなファイルになってしまいました。しかしその正体は，乱数で作った配列にすぎません。ですから，容量の限られた付録ディスクに入れるわけにはいきません。代わりに，rf. datを自動生成するプログラム，rfbuild. xを入れてあります。適当なドライブにrfbuild. xをコピーしておいて，

rfbuild

としてください。なにかメッセージが出てきますが，無視してください（実は，オプション指定で，ランダムフラクタルの形状を変えられるのです。しかし基本的に必要なさそうなので，オプションなしで実行してください)。そのうちにrf. datができてきます。それを，BドライブのZ's-EXと同じところにコピーしてください。以後はrfbuild. xは不要です。

それ以上の説明は不要でしょう。

第3章　使い方

（システムの起動）

Z'sSTAFFとZ's-EXのあるディレクトリ（前述のとおり，同じディレクトリにファイルを置いておいてください）に行き，そこでコマンドラインから，

Zs_EX

とします。あとは自動的にZ'sSTAFFが立ち上がります。先ほどのインストール手順のとおりにされた方は，AUTOEXEC. BATにここまでは書いてありますので，スイッチを入れてしばらくすると立ち上がるはずです。もし，グラフィックRAMを保存して立ち上げたいのであれば，この時点で，スペースキーを押しておいてください。

起動してしまえば，基本的にディスクドライブは自由です。PICファイルの入ったディスクに入れ替えても大丈夫です。ただ，文字入力をする際は辞書ファイルが必要なので，そのときにはAドライブを戻さなくてはならないでしょう。

また，Z'sSTAFFは，縦長の絵を編集するときに，巨大なテンポラリファイルを作ります。これはいまはBドライブに作っているので，どうしても縦長にしたいのなら，Bドライブも戻す必要があるでしょう。むろん，これらの機能を使わないのなら，起動してすぐにディスクを抜いてしまっても害はないと思われます。メモリが少ない場合はなおさらです。余分なものを組み込むべきでもありません。

起動した状態では，グラフィックもマウスもZ'sSTAFFの管理下にあります。お好きなように絵を描いてください。

（メインメニューの起動と終了）

Z's-EXシステムはZ'sSTAFFから呼び出しますが，その際ひとつだけ注意することがあります。

Z's-EXシステムを呼び出す前には，必ずZ'sSTAFFのすべてのウィンドウを閉じてください。ウィンドウ左上のタイトル部分をダブルクリックすれば閉じます（Z'sSTAFFを扱い慣れた人には，こんな説明は不要ですが)。メニューバー(起動時，画面の上端に出る長いもの）も消します。これにはスペースキーを押します。

これは厳守してください。それがもとで暴走するという性質のものではないのですが，画面がめちゃくちゃになってしまう可能性があります。なぜなら，Z'sSTAFFのウィンドウは，一部を除いてG-RAMに描いているからです。

Z'sSTAFFのウィンドウはすべて消す。これは絶対に忘れないようにしてください。

画面上からすべてのウィンドウが消えたのを確認したら，そこで初めてSキーを押してZ's-EXを呼び出してください。ビジュアルシェルのウィンドウに似た，Z's-EXのメインメニューウィンドウが現れるはずです。

ウィンドウ操作は，ビジュアルシェルとほぼ同様にしてあります。ウィンドウを動かしたいときは，タイトルバーをつかんでドラッグしてください。ドラッグはおわかりですね？　再びZ'sSTAFFに戻りたいときはメニュー左上のクローズアイコンをクリックしてください。メニューウィンドウが消えます。もうすでにここでZ'sSTAFFに戻っています。マウスをクリックすれば，Z'sSTAFFの動作をしますので気をつけてください。スペースキーを押せば，Z'sSTAFFのメニューバーが現れます。

メインメニューにはいくつか単語が並んでいます。これらはZ's-EXで使えるコマンドを表しています。そのなかから，使いたいコマンドを選んでクリックしてください。これから，各コマンドの説明に入ります。

（PICFILER）

ウィンドウの中央に，カレントディレクトリのファイルが表示されます。ただし，表示されるのはディレクトリとPICファイルだけです。ほかのファイルにはちょっかいを出せません。

ディレクトリを指定するときは，そのディレクトリの名前の左にあるボタン（ただの小さな四角ですが）をクリックしてください。そのディレクトリに移ります。ファイルを指定するときも同様の操作をします。つまりロードまたはセーブしたいファイルの名前の左にあるボタンをクリックします。すると，そのファイルの名前がウィンドウ下のファイル名表示エリアに現れます。

このファイルをロードするとしましょう。それには，ウィンドウの[LOAD]アイコンをクリックしてください。ロードするかどうか尋ねてくる（ちょっとうっとうしいのですが，誤操作防止のためです)ので，[OK]または[CANCEL]をクリックしてください。

もしセーブしたいなら，ウィンドウの[SAVE]アイコンをクリックすれば，ロードの場合と同様にしてできます。ただ，セーブの場合は，当然，新しくファイルを作ることが必要になることが多いのです。そのため，ファイル名の指定をキーボードか

PICFILER

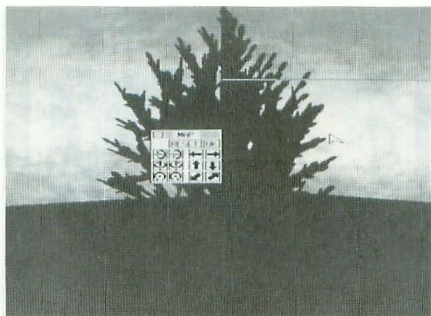
ら行うこともできます。その場合は，ウィンドウ下方のファイル名表示エリアをクリックしてください。ファイル名表示エリアにカーソルが点滅しますので，キーボードでファイル名を打ち込んでください。このとき，拡張子“.PIC”はつけてもつけなくてもかまいません。たとえ間違った拡張子をつけても，強制的に“.PIC”に変えます。

手抜きなことに，カーソルが点滅しているモードのあいだはマウス操作は効きません。間違えてファイル名指定モードに入ってしまったら，リターンキーを押して抜けてください。

ファイルウィンドウは8行しかありません。でもファイルやディレクトリが8つですむわけはありません。そこで，ファイルウィンドウをスクロールさせなくてはなりません。でも，ウィンドウのどこを見ても，スクロールアイコンはありません。ここでは，電脳倶楽部の標準シェルであるDSHELLで採用されて成功を収め，はたまた今回の付録ディスクのVS2（ビジュアルシェル拡張版）のドキュメントウィンドウでも採用されている操作方法を採用しました。

つまり，スクロールアップおよびスクロールダウンをするには，ファイルウィンドウ内で左および右クリックするのです。ただし，ファイルウィンドウ左端のボタンのところでクリックすると，すでに説明したとおり，そのファイルが選択されるだけですので気をつけてください（別に害はありませんが）。

ついでに，ファイルウィンドウの中で，左右のボタンを同時にクリックしてみてください。今回は左右クリックに，親ディレクトリに戻る“CD ..”の機能を持たせてみました。もちろん，ファイルウィンドウの“¥”ディレクトリを指定しても同じです。

ただ，これには異論のある方もいらっしゃることでしょう。DSHELLにしてもVS2にしても，左右同時クリックは「そこから抜ける」という意味です。その論法でいけば，PICFILERそのものを抜けてメインメニューに戻るというのが筋でしょう。ただ，DSHELLでは，左右同時クリックは「前の階層に抜ける」と解釈できなくもありません。それなら，階層化ディレクトリの前の階層に抜けるというこのPICFILERの作法もあながち間違いとはいいきれないかもしれません。DSHELLは，ルート（つまり，目次ですね）からは左右クリックで抜けられないようになっています。これは当然，誤操作でDSHELLを終わってしまわないようにするためのものでしょう。このへんも，ルートディレクトリからは抜けられないPICFILERと共通しているところですかね。どうしても我慢できないという方はご意見お待ちしております。もし次のバージョンを制作する機会があれば参考にさせていただきたいと思います。

これが微分処理

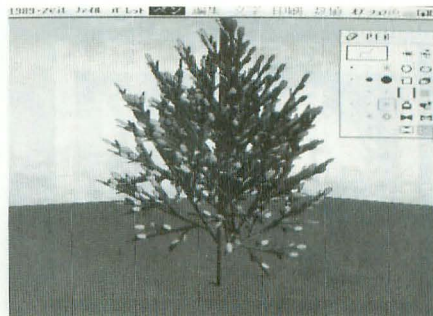
周りをマスクペイントしてマスクで指定

そしてマッピング

さて話が逸れてしまいましたが，お話し忘れていたことがひとつあります。ドライブを変える方法です。ウィンドウ左上に“A：”という表示のある部分，それがそうです。ドライブを変更するには，ドライブ表示部で左または右クリックしてください。フロッピーディスクの入っていないドライブはスキップします。のはずなのですが，ときどきシステムのほうから「ディスクが入っていません」というエラーを出されてあわてることがあります。Aドライブがフロッピーディスクで，なおかつディスクが入っていないときに出るようです。白状すると，プログラムを作るときはAドライブがハードディスクだったので，チェックを入れ忘れたのです。

左と右のボタンにまったく逆の機能を割り当てるという作法が，本ツールにおいては今後も頻繁に出現します。左でUP，右でDOWN……，まるでポピュラスですな。

メインメニューに戻りたいときは，ウィンドウ左上のクローズアイコンをクリックしてください。PICFILERのウィンドウが閉じ，メインメニューが現れます。

(ALTERNATE SCREEN)

裏画面と作業画面を切り替えます。説明の必要もないでしょう。

(MAPPING)

MAPPINGを呼び出すと，十数個のアイコンがつまったウィンドウが開き，画面には妙な枠が出てくることでしょう。試しに，下半分に並んでいる，中に図の描いてある四角いアイコンをいろいろとクリックしてみてください。枠がぐるぐると回ったりずるずると動いたりすることでしょう。左ボタンだけでなく，右でもクリックしてみてください。

左半分の，棒の周りに矢印が回っているようなアイコンは，「座標軸まわりの回転」を，右半分の，矢印が描いてあるアイコンは「座標軸方向の移動」を，それぞれ表しています。

座標軸は，上からx，y，zの順番になっています（これは，そう意識する必要はありません。枠はアイコンに描いてあるとおりに動くようにしているつもりです）。

左クリックは表示してあるとおりの方向に，右クリックは表示してあるのとは逆の方向に移動および回転を行います。これまた右に機能を持たせてしまったのは，ある意味では迷惑なのかもしれません。が，この手の作業では往々にして「あら動かしすぎたよ」というミスが発生しがちです。そんなときにマウスをもう一方のアイコンのところに持っていって左クリックでもよいのですが，それはなかなか面倒。それよりも空いているボタンで逆の動作をすぐにできるようにしたほうがありがたいのではないかと思ったわけです。

さて，ぐるぐる動かしているうちに，なにがどうなってしまったのかわからなくなることがあります。枠にはひとつ印をつけてあるので，どんなになっても方向はわかりますが。また，マッピングする画像を新しいものにかえた場合など，いつまでも前の結果を残しておくのは気持ち悪い。で，そんなときは，RESETアイコンをクリックしてください。初期状態に戻ります。枠

光る部分をマスクで指定

フレア処理するとこうなる

は画面一杯，移動も回転も1回もしていない状態に戻ります。新しい気持ちで動かし直してください。

いよいよ絵を張り付けることにします。裏画面の準備はいいですか？　裏画面にマッピングするデータを用意しておかないとなにもなりません（というよりも，表画面に真っ黒な絵が張り付くだけになります)。準備ができたら，OKアイコンをクリックしてください。

実行は，はっきりいって速くありません。しかし，起動してから初めてMAPPINGに来たとき，それから回転も移動も行っていないとき，移動・回転をRESETしたとき，それから回転も移動も行っていないとき，つまり枠が初期状態にあるときに張り付けると，特別に高速で処理してくれます。これは使えます。画面合成が数秒で行えます。

メインメニューに戻りたいときは，PIC FILERと同様に，ウィンドウ左上のクローズアイコンをクリックしてください。

(MASK PAINT)

これを選択すると，メインメニューは閉じます。マスクをかけたいところにマウスカーソルを持っていって，左クリックしてください。その部分にマスクがかかって，青く点滅しはじめます。メインメニューに戻るには，右クリックします。

なお，マスクペイントをかけてからZ'sSTAFFに戻ると，ときどきマスクがうまくかかっていないように見えるときがあります。ウィンドウを動かしたあとに，塗ったはずのマスクがなくなっているときがあるのです。原因はよくわかりません。こんなときは，あわてずに，Z'sSTAFFの「ペン」メニューの中のマスク反転アイコンを2回使ってください。それで正常になるようです。

(EFFECT)

サブメニューウィンドウが開いて，その中に4つアイテムが並んでいます。それらのうちひとつを選択してください。するとウィンドウが消えます。エフェクトをかけたい領域をマウスで指定してください。エフェクトをかけたい矩形領域の左上と右下にマウスカーソルを持っていって，左クリックしてください。その中に選択したエフェクトをかけます。指定を間違えたときは右クリックでやり直せます。メニューに戻るときは，1点も指定しない状態で右クリックしてください（要するに，どんな場合でも2回右を押せば帰ってきます)。

白黒処理(MONOTONE)と微分処理(DIFFERENTIAL)については説明の必要はないでしょう。一度試せば理解できます。問題は残りの2つ，ランダムフラクタル処理(RANDOM FRACTAL)とフレア処理(FLARE)です。この2つには，パラメータがあるのです。

(RANDOM FRACTAL)

囲んだ領域を自然なランダムさで（なんじゃそりゃ）崩します。ランダムフラクタル処理は，グラデーションのような，滑らかに変化する画像に対して威力を発揮します。縦にグラデーションをかけた部分を囲んでみてください（アルゴリズムの性質上，横方向に変化するグラデーションを処理しても意味がありません)。雲のような画像が現れるはずです。

さて，メニューウィンドウのRANDOM FRACTALと書いてある右に，数字の入った枠があります。これがパラメータです。この値は，フラクタルで画像の崩れる度合いを示したものです。ここでマウスを左または右クリックすれば，パラメータの値が増減できます(左で＋，右でー)。値の取りうる範囲は0から9のあいだです。値が大きいほど崩れ方も派手になります。

(FLARE)

光源にしたい部分をマスクしておき（これは前述のマスキングペイントでもいいし，Z'sSTAFFのマスキングでもいいのです)，その周りを囲みます。すると，マスク部の周辺に，その光源の色が淡く滲み出てくるはずです。そこでマスクを解除すると（これにはZ'sSTAFFのマスク解除命令を使ってください)，光源がぼうっと光っているように見えることでしょう。

フレアにもランダムフラクタルと同様に，ウィンドウ内にパラメータの入った枠が2つあります。フレアの場合パラメータは2つです。左のパラメータは明るさ，右のパラメータは広がりぐあいを表しています。

フレアは，扱いに慣れるまで多少時間が掛かると思います。理想をいうなら，光を滲ませるアルゴリズムを理解したうえで使うのがいいのです（気分としてもっとも近いのは「ぼかし」処理です)。こつとしては，内側のマスクより大きめに範囲を指定することがまず挙げられるでしょう。でないと，光の広がりが途中で切れます。これはみっともないものです。また，マスク部分の形や大きさによっても，フレアの様子はかなり変わってきます。うまく使うと面白い効果が出せますので，しばらく使ってみてぜひとも体得してください。

＊

ものによっては，非常に実行が遅い機能もあります。そんな機能はたいてい，右ボタンを押しているうちに実行を中断して戻ってくるようにしてあります。

例によって，ウィンドウ左上のクローズアイコンでメインメニューに戻ることができます。

第4章　おわりに

今回は難しい話は抜きにしました（おいおい本当かよ)。無事に使えるようになりましたか？

メモリが1Mバイトの方は増設してください。ただ，Z's-EXはZ'sSTAFFから起動しなくてもエフェクタとして使えるプログラムですから，うまいプログラムを作って名前をSTAFF68K.Xにしてしまえば，1Mバイトでも動くようになるかもしれません。気が向いたらフォローします。

しつこいようですが，Z'sSTAFFのウィンドウをすべて閉じてからZ's-EXを起動してください。そこそこ使えるツールだと自負しております。ぜひとも活用してください。

あと，操作性や機能で気にいらないところ，ほかにこんな機能もほしいといった要望などございましたら，お知らせいただけると幸いです。

最後に，このプログラムを実現する可能性を開いてくださった，電脳倶楽部とPIC FILERの山田浩示さんにお礼を申し上げたいと思います。

グラフ作成ツールMichael

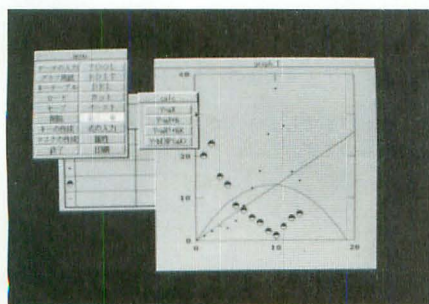
ディスクでしか掲載できないちょっと大きな投稿作品、グラフ作成ツールMichaelです。「マイケル」と呼んでください。基本的なグラフ作成機能に加え、グラフィックツールの要素を盛り込んでいます。なお、マルチウィンドウですがSX-WINDOW上のソフトではありません。ご注意ください。

Ueno Kazuhiko
上野 和彦

いちいち紙にグラフを描いてデータの傾向を見るのは面倒だ，という実験屋のあなたにピッタリのソフトを作ってみました。片対数/両対数用紙もサポートした簡易グラフィックエディタつきのグラフ作成ツールMichaelです。マルチウィンドウの採用で画面上でグラフを紙のように扱えます。グラフの印字品質もまあまあで，市販のグラフ作成ツールのものと比べても見劣りしません。3次元グラフとか棒グラフといった凝ったグラフは表示できませんが，私たちがふつうにグラフ用紙にX軸Y軸をとって描くようなグラフについては自信を持っておすすめできます。

ただし，プログラムの性格上，プリンタを持っていないとほとんどありがたみはありません。CZ系の24ピン(または48ピン)プリンタをお持ちの方，実験のデータ整理などにぜひお役立てください。

Michaelとは

このツールは「人間がグラフを描く動作をパソコンでシミュレートする」ということをイメージして作りました。私たちが日常グラフを描くときの動作を思い起こしてみましょう。まず，座標軸をとり，対応する点を打ってその上から定規で直線を引いたり，自由定規で曲線を引き，テンプレートで文字を入れていきます。このツールはこれとまったく同じ手順でグラフを描いていくものです。

これがMichaelだ

X，Yの座標を入力すると点を打ってくれて，そのあとグラフィックエディタで好きなところに線や文字が入れられます。このため，表現力豊かなグラフが作成できます。どんな強力なツールを作っても結局最後は人間の手で文字や図を入れたくなるんですよね。

以下にMichaelの特徴をまとめてみましょう。

1)　マルチウィンドウの採用で同時に最大4つまでのグラフが扱える
2)　グラフィックエディタがついているので，簡単に文字や直線，曲線が入れられる
3)　最小2乗直線の計算ができる
4)　キーが豊富である
5)　印字の際は画面イメージの1.5倍のフレームを取るので印字品質がよい
6)　各グラフには説明がつけられるため，あとで見てなんのグラフかすぐにわかる
7)　グラフの表示範囲を自動的に設定する(もちろん，あとで修正できる)

使用方法

Michaelを使用するには少なくともプリンタドライバと数値演算ドライバが組み込まれた状態で，Michaelのあるディレクトリに移動したうえで，

A>MICHAEL WORK¥

のように起動してください。

付録ディスクの場合，ディスク2のMichael.BATのアイコンをダブルクリックすれば起動できます(ただし印刷などは不可能)。

ウィンドウ部の基本操作

●ウィンドウの移動

ウィンドウの上側についているタイトルバーを1回クリックして，移動したい位置でもう一度クリックします。

ビジュアルシェルやSX-WINDOWのようなドラッグ操作ではありませんので注意してください。このツールではマウスをドラッグして操作するということはありません。あと，クリック操作は軽快に行ってください。これさえ覚えておけば，あとはだいたい勘でわかるでしょう。

●ウィンドウ内のスクロール

グラフ用紙を大きくしたときスクロールバーが現れます。非常に細いので注意して見てください。そこをクリックするとウィンドウの中身をスクロールさせることができます。

その他の操作については図を参照してください。

コマンド解説

起動時に表示されるメニューウィンドウに沿って各コマンドを解説していきます。コマンドを実行するときはメニューウィンドウの各ボタンをクリックしてください。

1)　データの入力

グラフにプロットする点の座標を入力するコマンドです。メニューウィンドウの，<データの入力>というボタンをクリックするとテーブルウィンドウが開きます。1から4までのうちどれかを選択すると対応する表ウィンドウが開きます。

図2を見てください。たとえば，(1.0, 2.0)，(1.1, 2.1)，(1.2, 2.2)，(1.3, 2.3)の4点をグラフにプロットしたい場合，まずX座標だけを横1行に入力して，次にY座標のみを別の行に入力します。

リターンキーを押すとメニューに戻るので注意してください。場所の移動はカーソルキー(コントロールキーならさらによい)を使用してください。キー操作は，

CTRL+E	上
CTRL+D	右
CTRL+X	下
CTRL+S	左
CTRL+H	バックスペース
CTRL+M	リターン
CTRL+Y	1行カット

CTRL＋L　1行ペースト
CLR　　初期状態に戻す

となっています。

X座標をどの行に書いたかをMichaelに通知しなければなりません。X座標を書いた行をクリックするとキーウィンドウが開くので，ここからXのマークを選びます。同様にプロットするデータの記号を選択してください。

2)　グラフ用紙

表示するレンジ，グラフの大きさ，目盛りの種類などを決めます。

データ入力後，メインメニューから<グラフ用紙>を選択し，適当に設定するとグラフが表示されます（グラフ用紙設定から抜けるにはメインメニューまたはデータ入力ウィンドウをクリックしてください)。データの入力が不完全な場合はグラフは表示されません。注意してください。

●目盛り

対数設定ができます。Xを普通にYを対数にすると（逆でもいいんですけど）片対数グラフになります。目盛りと書いてある位置のボタンを押すと表示が交互に切り替わります。目盛りを変えると表示レンジも変えなくてはいけないのでレンジも自動設定します。また，座標中に0または負の数があると対数に設定できません。

●最大値，最小値

グラフの目盛りの最大値と最小値です。最初は自動設定された値になっています。対応するボタンをクリックすると入力ウィンドウが開くのでそこで入力してください。キーボードがローマ字や全角のモードになっていると設定できません。

●数字間隔

数字を表示する間隔です。対数目盛りのときには10と1しか設定できません。1にするとすべての目盛りに数字を打ちます。10にすると10目盛りおきに数字を打ちます。

●目盛り間隔

目盛りを刻む間隔です。対数目盛りのときには10と1しか設定できません。

●説明

X軸，Y軸の説明です。対応するボタンをクリックすると入力ウィンドウが開きます。日本語FEPを組み込んでいるときは日本語入力もできます。

●幅

グラフの表示ドット数です。この数字を変えると印刷したときのグラフの大きさも変わります。

3)　キーテーブル

キーテーブルの表示位置を決めます。ウ

図1

ネームバーの左右にあるマークは，Michaelでは表にしかついてません。表を閉じると，それに対応するグラフも同時に閉じるようになっています。

スクロールバーは，グラフウィンドウが大きくなったときに現れます。小さいので見落とさないようにしてください。ウィンドウの大きさは，今回はMichaelに制御を任せて固定にしました。

図2

ファイル名：ここをクリックするとファイル名の入力が出来ます。
説明　　　：ここをクリックすると表の説明が入力出来ます。

		1	2	3	4
X	x_data	1.0	1.1	1.2	1.3
○	y_data	2.0	2.1	2.2	2.3
●	y1_data	3.0	3.1	3.2	3.3

図3

キーの作成もマスクの作成も操作は同じです。24ドットのエディットエリアでパターンを書いて，16ドットに変換して，16ドットエディットエリアで変換後のパターンを修正するという手順がよいでしょう。

キーはわかるけど，マスクってなんだ？　と思う方もいるでしょう。マスクは，キーの型紙です。つまり，キーを表示する部分を黒く塗ったものです。

ィンドウ外に設定すると表示しません。

4) ロード

表をロードします。ファイルウィンドウにファイル名が表示されるので目的のものをクリックしてください。

5) セーブ

表をセーブします。複数の表を開いているときには，どの表をセーブするかを選んでください。

6) 削除

ファイルを削除します。ファイルウィンドウで削除するファイルを選んでください。

7) キーの作成

グラフに点をプロットするときのキーを作成します。

8) マスクの作成

グラフに点を表示するときのマスクを作成します。

9) 終了

Michaelを終了します。

10) TOOL

グラフをエディットするときのモードを決めます。

●エディットモード

直線，曲線，文字から選んでください。

●線種類

エディットモードで直線，曲線を選んだとき描画に使用する線の種類を決めます。線の太さが2つと実線，点線，1点鎖線，2点鎖線の4つの線種，合計8パターンが使用できます。が，画面上では線の太さは区別がつきません。線の種類で区別することをおすすめします。

●文字種類

エディットモードで文字を選んだとき描画に使用する文字の大きさを決めます。大文字と小文字のどちらかを選んでください。

●グリッド

カーソルの動きを8ドットおきにします。一定間隔で文字を入力するときや平行線を引くとき便利です。

11) EDIT

グラフをエディットします。表が複数開いているときはどの表のグラフをエディットするか聞いてきます。また，グラフウィンドウをカレントにした場合もこのモードとなります。

12) DEL

エディットで入力した図，または15)，16)で入力した式などを消します。消したい線や文字をクリックしてください。このコマンドから抜けるときはグラフウィンドウの外をクリックしてください。

13) カット

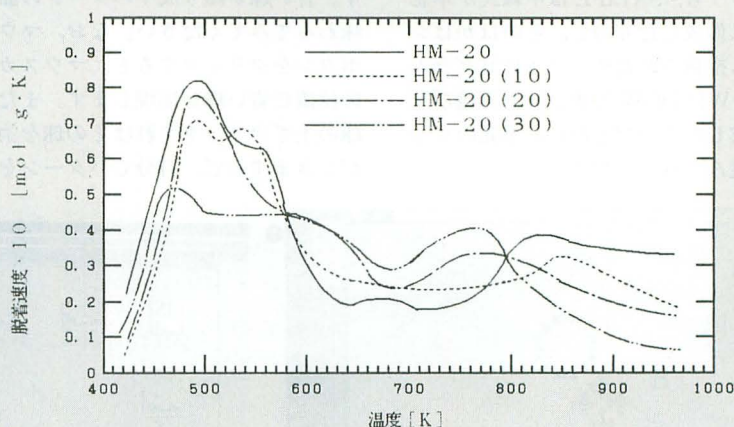
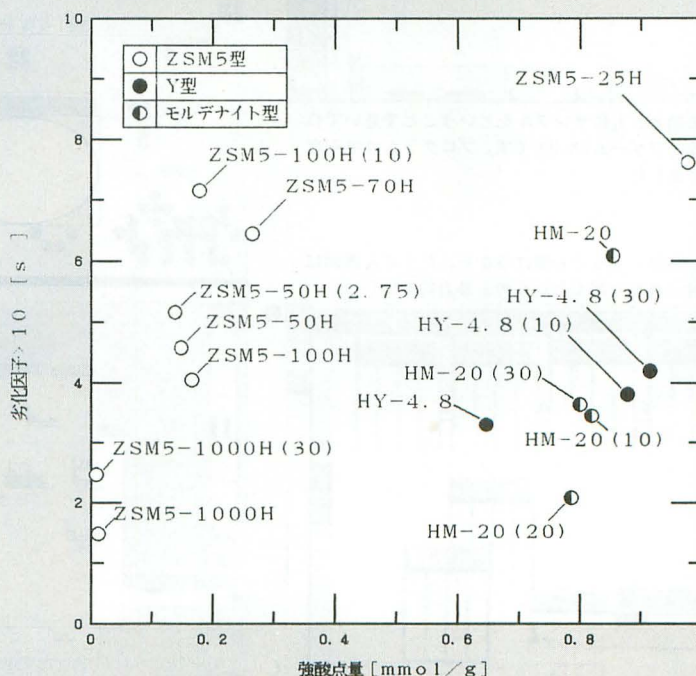
エディットで入力した図，または15)，16)で入力した式などをすべて消します。

14) ペースト

13)でカットしたデータを復帰します。

15) 最小2乗

表に登録されているすべての点に対して最小2乗法によって回帰曲線を求めます。Y＝aXは化学系の検量線の作成などに便利だと思います（必ず原点を通らなければならないときはY＝aX＋bが使えないんですよね）。

Y＝bEXP（aX）は片対数グラフの直線です。この型はいろんな式に出てくるのでそれらの定数を求めるのに便利です（グラフも表示されているので，はずれたデータを除いて最小2乗を取ることができます）。

また，このときの線種はTOOLで指定したものとなります。

16) 式の入力

限られたかたちしかありませんが式を入力してグラフにできます。

17) 属性

最小2乗で計算した定数などを表示させるときに使います。グラフの特定はDELとほぼ同じです。

18) 印刷

グラフをプリンタに出力します。用紙はB5を想定しています。プリンタのA4の位置に設定してください。

＊　＊　＊

それでは皆さん，ぜひこのツールを活用してください。

図4

SX-WINDOW開発セット&アクセサリプログラム

解説　泉 大介
資料提供／シャープ株式会社

SX-WINDOWの内部に興味がある人，プログラムを組みたいと思っている人には，なくてはならない技術資料，ツール類，Cライブラリを一挙公開です。誌面には絶対収まらない膨大な量ですが，ぜひとも活用してください。いくつかのサンプルプログラムも楽しめます。

ディスク3のアクセサリ

ディスク3のSX_samplというディレクトリにはSX-WINDOW上で動くちょっとしたアクセサリプログラムが入っています。このうち，SXLIFEは中森氏が本誌の記事用に作成したもので，そのほかはシャープから提供されたサンプルプログラムです（SX-WINDOWのチェックと遊びを兼ねて作成したものだとか)。お手元のウィンドウで遊んでみてください。

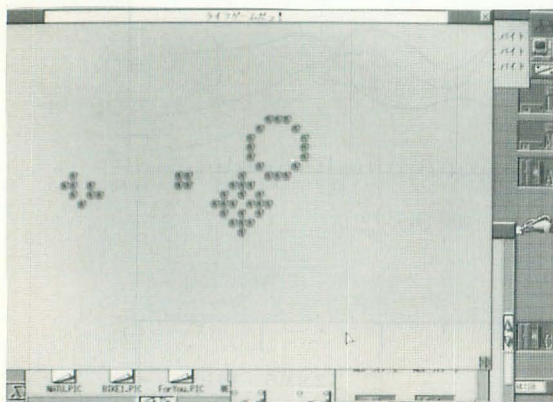
●SXLIFE

SX-WINDOW用のライフゲームです。それぞれの青い球の周りに何個の青い球が存在するかで，その青い球が次の世代に生き残れるか，死んでしまうか，それとも新しい青い球が誕生するかどうかが決まります。青い球が織り成すパターンの面白さを味わってみてください。なお，マウスの右ボタンをクリックすると，マウスカーソルの位置に青い球が出現します。また，青い球の上でクリックすればその球を消すことができますので，自分でパターンを置いてその変化を楽しむこともできます。

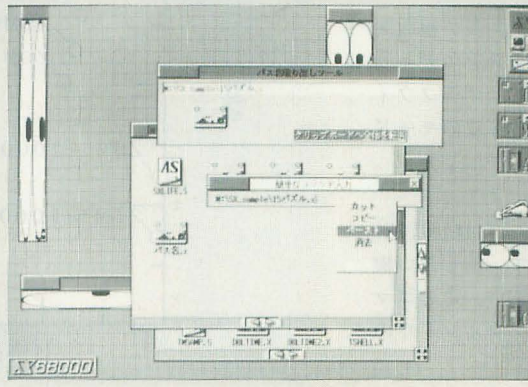
●SXeyes

目玉がマウスカーソルを追いかけるというサンプルです。ウィンドウシステムには定番のアイテムですね。メモリに余裕のある方は，目玉を10個も表示すれば一種異様な雰囲気に浸ることができます。ぜひお試しを。

●15パズル

4×4のマスに入っている15個のタイルに番号が振ってあります。それを1から順に並べ替えてください。16個のマスに15個の

❶ ▲技術資料の公開とともにサンプルをということで急いで作ったのがこのライフゲームSXLIFEです。プログラムは中森章さんにお願いしました。

❷ ▲ライフゲームをバックに敷き，15パズルに興じるの図。目玉も注目しているぞ。

❸ ▼15パズル.X。当然いくらでも開けるから，たくさん画面に出しておいて片っ端から解くというのも暴力的でいい。

◀パス名.X。アイコンを放り込むとパス名を表示し，クリップボードに取り込める。長いフルパス指定をするのに便利。

タイルですから１マスの空きができます。空きマスと同じ行，あるいは同じ列のタイルの上で左ボタンをクリックすると，クリックされたタイルが空き升の方向に押しやられるようにして移動します。こうしてタイルを移動しながら数字を揃えていくわけです。

●パス名

ファイルのフルパス名を表示するウィンドウを開くサンプルです。フルパス名を表示したいファイルのアイコンを，ドラッグしてこのウィンドウに放り込んでください。「クリップボードへ全体を転送する」と書かれたボタンをクリックすると，フルパス名がクリップボードに転送されます。転送されたパス名は，ノートなどのプルダウンメニューで取り出すことができます。このプログラムはもともと，ノートでCONFIG. SYSを書き換えるために作られたものでした。

＊

本体メモリが2Mバイト以上の方は，なにも問題はありません。SX-WINDOWを起動したあとディスク3，ディスク4を挿入し，サンプルプログラムをダブルクリックしてください。本体メモリが1Mバイトしかない方は，メインメモリ1Mバイトで起動するSX-WINDOWのシステムディスクの作り方を囲みで紹介していますので，1Mバイト用のSX-WINDOWシステムディスクを作成してサンプルを実行してください。

ディスク4の内容

ディスク4はSX-WINDOW用アプリケーションの開発ディスクです。これらはシャープから提供されたものを編集部で構成したものです。ある程度の技術的知識を持った人を対象とした最小限の資料ですから，十分な理解は困難かもしれません。本誌では，今回の特集をはじめ，今後ともSX-WINDOWの解説を行っていく予定です。また，くれぐれもメーカーに直接問い合わせることはご遠慮ください。

SXディレクトリの中には，開発ディスクのファイル内容を説明するREADME. DOCと，6つのサブディレクトリが収められています。README. DOCは都合上ディスク3に収めたアクセサリプログラムの説明も兼ねており，README. DOCでQuickStartディレクトリ内のファイルとして説明されているのがディスク3のSX-Sampleディレクトリに収められているプログラムです。なお「とけい. X」は入っていません。ご了承ください。

このディスク4に入っているファイルは大きく3つに分けられます。ひとつは開発のための資料，もうひとつは開発のためのツール類，そしてSX-WINDOW用のサンプルプログラムです。このディスクはSX-WINDOW用のアプリケーション開発の補助ディスクであり，これ単体でアプリケーションが作れるというものではありません。XCあるいはアセンブラ，リンカが別途必要となります。すでに発売されているこれらの開発言語を，SX-WINDOW用のアプリケーションが開発できるように拡張するものと思っていただければいいでしょう。

サンプルプログラムについて

初めに，ディスク4に入っているサンプルプログラムの簡単な使い方を紹介しておきます。

● GRSAMP

グラフマンの機能を使って，画面に図形や文字を表示する小さなサンプルです。

● TSSAMP

ウィンドウを開いて文字を表示するという，最も基本的なプログラムです。SX-WINDOWプログラミングの基礎を学ぶにはちょうどいいサンプルだと思います。

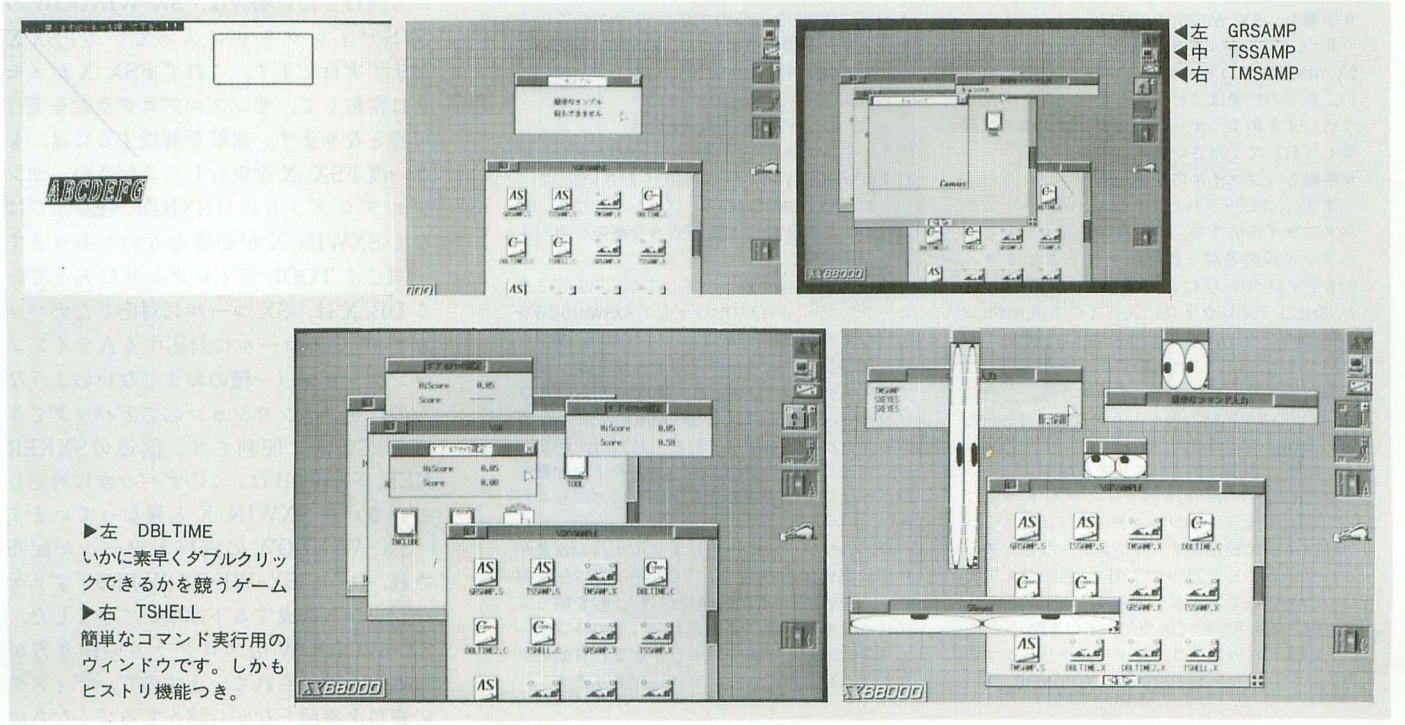
● TMSAMP

コマンド実行用のウィンドウです。COMMAND. Xと同じようにコマンド名をキーボードから入力し，リターンキーを押してください。このウィンドウでは，COMMAND. Xのコマンドを入力しても実行することはできません。

● DBLTIME

ダブルクリック間隔の短さを競うゲームです。最も短い時間がハイスコアとして表示されます。ちなみに編集部での最短タイムは0.03秒です。逆に，ダブルクリックの間隔の長さを競うゲームとしても遊べます。

◀左　GRSAMP
◀中　TSSAMP
◀右　TMSAMP

▶左　DBLTIME
いかに素早くダブルクリックできるかを競うゲーム
▶右　TSHELL
簡単なコマンド実行用のウィンドウです。しかもヒストリ機能つき。

こちらはハイスコアは残りません。

このプログラムはウィンドウ間の通信サンプルとして用意されたものです。複数のDBLTIMEを起動すると，スコアはそれぞれのウィンドウで独立ですが，ハイスコアは通信によって他のDBLTIMEのウィンドウにも伝えられ，常に最速のタイムが表示されます。

● DBLTIME2

DBLTIMEをリエントラント（再入可能）に書き直したものです。

● TSHELL

TMSAMPの機能を拡張したものです。こちらはHuman68kのコマンドも実行することができ，さらにヒストリ機能を備えています。コマンド入力の履歴はカーソル上下キーでたどることができます。また，カーソルキー，BS・DELキーなどを使って，以前入力したコマンドを再編集して実行することも可能です。

SX-WINDOWに関する資料

開発資料の中でまず目を通したいのは，DOCディレクトリに収められたファイルです。ビジュアルシェル上（VS2.X）でもダブルクリックで読むことができますが，積極的に活用するにはED.Xで読み込んだり，プリンタに打ち出すとよいでしょう（数100ページになりますが)。また，WP.Xのファイル入力でも読めますが，GRAPH.DOCなどはファイルサイズが大きすぎて一度には読み込めません。

ここに収められたファイルは，SX-WINDOW上のアプリケーションの動きを制御する各種マネージャを理解するための資料です。低レベル（ハードウエアに近いレベル）のマネージャから順に並んでいます。高レベルのマネージャのほうがアプリケーションと直接関わり合うのですが，SX-WINDOWで使われる用語の理解のためにも最初から読み進んだほうがいいでしょう。

SX-WINDOWの各マネージャとのやりとりは，SXコール（Human68kのDOSコールのようなもの）を利用して行います。このSXコールのリファレンスマニュアルが収められているのがREFディレクトリです。SXコールはここにマネージャごとにまとめられています。ファイルの拡張子がREFとなっていますが，DOCファイルと同様に内容は普通のドキュメントファイルですので，ED.XやWP.Xで読むことができます。

ここには，SXコールの一覧を収めた拡張子がLSTのファイルも入っています。DOC，REFファイルはすべてプリントアウトすると400ページにも及ぶ大作ですが，LSTファイルはコンパクトですのでプリントアウトして綴じておくと便利です。

開発を支えるファイル群

SX-WINDOW用のプログラムはアセンブリ言語，またはC言語で作成します。アセンブリ言語を使用する場合は，INCLUDEディレクトリのSXCALL.EQU，SXCALL.MACを利用します。これらのファイルにはSXコールのエントリとマクロが定義されています。

C言語を利用する場合は，INCLUDEディレクトリに入っているヘッダファイルと，LIBディレクトリに入っているライブラリを使用します。SXコールと該当するCの関数はREFファイルで対にして解説されています。

TOOLディレクトリに入っているSXKERNELは，SX-WINDOW本体であるSXWIN.Xからファイル操作やディスク操作といったデスクトップ機能を省いたものです。これはSAMPLEディレクトリに入っているTSSAMPなどをコマンドラインから実行すると起動されます。プログラム開発途中での，実行・チェック用に用意されたものだといえるでしょう。

TSSAMP.Xを実行するにはFSX.Xがメモリに常駐している必要があります。手持ちのHuman68kのCOMMAND.Xから実行したい場合は，SX-WINDOWのSYSディレクトリに入っているFSX.Xをまず実行します。これでFSX.Xがメモリに常駐して，サンプルプログラムを実行可能となります。常駐を解除するには，もう一度FSX.Xを実行してください。サンプルプログラムにはSXKERNEL.XではなくSXWIN.Xが必要なものもあります。

同じくTOOLディレクトリに入っているDB.Xは，SXコールに対応したデバッガです。SXコールに対応するAラインファンクション（一種のおまじないのようなもの）をファンクション名でデバッグできるので非常に便利です。前述のSXKERNEL，SXWDBは，このデバッガに対応している点がSXWIN.Xと異なっています。

SX-WINDOWの資料とツールが配布され，やっとSX-WINDOW用のアプリケーションを作成する下準備ができました。SX-WINDOW用プログラムの作り方が特集でも触れられていますので，ディスクの資料を参照しながら読んでみてください。

1MマシンでSX-WINDOWを動かす

1MバイトのX68000でSX-WINDOWを起動する方法を以下に示します。

●手順1　SX-WINDOWのコピー

ビジュアルシェルやコマンドモードなどでSX-WINDOWのコピーディスクを作ってください。以下の作業はコピーされたディスクを使って行いますので，オリジナルディスクは再び保管しておいてください。

●手順2　ファイルのリネーム

まず，(コピーされた)SX-WINDOWのディスク内のファイルのうち，大きなメモリを必要とするファイルの名前を付け替えてしまいます。SYSディレクトリに入っているASK68K.SYSと，SHELLディレクトリに入っているHENWIN.Xの2つのファイルを，適当な名前に変えてしまってください。これらのファイルが占有しない分だけフリーエリアが増えますから，これでSX-WINDOWが起動できるようになります。ただし，漢字は使えません。

●手順3　さらにフリーエリアを増やす

変更したSX-WINDOWのディスクをドライブ0に差し，リセットボタンを押してください。SX-WINDOWが起動します。カレンダーなどが表示されていることと思いますので，右上のXアイコンのところで右ボタンを押し，「全クローズ」を選択して一旦すべてのウィンドウを閉じます。

Aドライブのアイコンをダブルクリックすると表示されるウィンドウの中からCONFIG.SYSを探してください。見つかったらこれをクリックして反転表示にし，次にノートを開きます。ノートにCONFIG.SYSの内容が表示されていますね。ノートの中央あたりにある，

```
DEVICE=¥SYS¥PRNDRV.SYS
DEVICE=¥SYS¥RSDRV.SYS
DEVICE=¥SYS¥OPMDRV.SYS
```

の3つの行の先頭に'*'を付け，saveボタンを押してCONFIG.SYSをセーブしてください。

これはSX-WINDOWから，プリントアウト，RS-232Cによる通信，FM音源の演奏機能をカットする指示です。

ノートのsaveボタンを押してCONFIG.SYSをセーブしたら，再びリセットしてSX-WINDOWを起動してください。FM音源が鳴らない，漢字が使えないなど機能縮小版ではありますが，これで1Mバイトのメモリでも支障なくサンプルプログラムを実行できるSX-WINDOWシステムディスクができあがりました。ただし，相変わらずメモリは苦しいので，SXeyesを2つ起動するようなことはできません。順々にお試しください。

今回のサンプルにはありませんが，FM音源が必要な場合は再びノートでCONFIG.SYSを読み込み，OPMDRVの行の先頭に付けた'*'を削ってリセットすればOKです。ただし，その分フリーエリアは減りますので，もしかすると目的のプログラムが実行できない可能性があります。

特別付録　謹賀新年PRO-68K

ウイルス検出プログラム DOCTOR2.X

編集部　プログラム協力：中森　章

DOCTOR2.XはX68000上のウイルスを治療または予防するワクチンプログラムです。使用上の注意事項をよく読んでお使いください。また，旧バージョンのDOCTOR.Rをご使用の方は今回のDOCTOR2.Xに差し換えてお使いください。

DOCTOR2.Xは，X68000のSRAMに感染するウイルスに対応するワクチンプログラムです。本誌で確認されているX68000上のウイルスはいずれもSRAMに入り込むことで感染するというもので，フロッピーディスクに記録されるIPL部分を書き換えることで，次々と別の本体に感染していく性質を持っていました。そこで，このDOCTOR2.Xは主に次のような機能を持たせています。

1)　フロッピーディスクのIPL領域を検査，および治療する。
　　DOCTOR2
2)　本体のSRAMを初期化する。
　　DOCTOR2 －P
3)　SRAMにDOCTORを常駐させる。
　　DOCTOR2 －T

＊なお，DOCTOR2 －? などでオプションスイッチのヘルプが出ます。

それでは，基本的な使用法について具体的に解説しておきましょう。

フロッピーディスクの検査と治療

ドライブ0に挿入されたフロッピーディスクのIPLを検査します。手持ちのシステムディスクを起動し，調べたいフロッピーディスクをドライブ0に挿入してください。

DOCTOR2.Xの入ったディスクをドライブ1に入れ，

A>B:DOCTOR2

と実行してください。異常がなければ，

```
IPLをチェックしています。セルフチェック OK
これは Human 純正の IPL です。
ドライブ0の Human は正常です。
```

と表示されるはずです。

ただし，ゲームソフトなどの一部でオリジナルのIPLを使用しているものもあります。その場合には，

```
これは Hudson 純正の IPL ではないが大丈夫でしょう。
シフトキーを押すと続行します。
```

となります。

もしも，ウイルスに感染するなどの原因で正常なIPLが書き換えられている場合には，画面が一瞬赤くフラッシュし，

```
これはウイルスに冒されています!!!!!

注射をしますので，ドライブ1に健康な Human Ver 2.0(相当品)を入れて，
リターンキーを押してください。
```

といったメッセージを表示します。注射というのは，冒されたIPLの部分に正常なIPLをコピーするということです。ただし，問題となったいわゆるFORCEウイルスの場合にはIPLのバージョンが書かれた部分も書き換えられるため上記のメッセージが“Human Ver Pro”などとなってしまいます。純正のIPLはHuman68kのバージョンによって，1.0と2.0の2種類がありますが，いずれのバージョンでも特に支障なく治療できるはずです。メッセージにしたがって安全なシステムディスク（マスターディスクなど）をドライブ1に入れてリターンキーを押してください。

```
ドライブ0の Human は健康になりました。
よかったね。
```

となればOKです。ここで，治療しようとしたディスクにライトプロテクトシールが貼られていたりすると，

```
ドライブ0の書き込みに失敗しました。(書き込み禁止?)
プロテクトシールをはがしてドライブ0にディスクを入れ直し
リターンキーを押してください。
```

と表示されます。再度確認してください。

さて，注意してほしいのは，IPLが純正ではなくオリジナルのIPLの場合です。もともとプロテクトシールが貼ってあるゲームソフトの場合ならほぼ安全ですが，万一オリジナルIPLを使用したディスクがウイルスに感染した場合，DOCTOR2.Xでも治療はできません。ウイルスは殺すことができますが，起動できなくなってしまいます。

SRAMを初期化する

2番目の機能は，SRAMの初期化です。ウイルスに感染したフロッピーディスクでシステムを起動すると，ウイルスはSRAMに入り込み，X68000が感染した状態になります。そこで，SRAMのプログラムエリアを初期化するには，コマンドモードから，

DOCTOR2 －P

としてください。

これで，SRAMのプログラムエリア($ED0100～$ED3FFF)が初期化されます。ただし，SRAMを使用中の場合，内容が消えてしまうのでご注意ください。

また，ウイルスの感染によってシステムがうまく起動しなくなったときは，OPT.1キーを押しながらシステムディスクを起動させ，SRAMの初期化を行ってください。

DOCTORをSRAMに常駐させる

上記の2つの機能で，ウイルスに感染したディスクやマシンを治療することができます。しかし，できることならウイルスへの感染は未然に防ぎたいものです。

そこで，DOCTORにはあらかじめSRAMに常駐させて起動時のチェックを行う機能があります。コマンドモードから，

DOCTOR2 －T

としてください。これでDOCTORがSRAMに転送されます。

以後，フロッピーディスクで起動するたびに自動的にディスクのIPLを調べ，上記のDOCTOR2を実行した場合と同様のメッセージを出すようになります。

ドライブ0にディスクが入ってないと，フロッピーの挿入を待つメッセージを出します。

```
ドライブ0に Human のディスクを入れてください。9
```

ハードディスクがつながっている場合には，カウントが0になるまで待ってハードディスクからの起動に切り替わります。が，最初からハードディスクで立ち上げたい場合にはシフトキーでカウントダウンを中止してハードディスクから起動します。

なお，DOCTORを外したい場合には，

DOCTOR2 －P

で，SRAMを初期化するだけでOKです。

補足解説

今回のDOCTOR2.Xは1990年6月号で発表したワクチンDOCTOR.Rを新たに作り直したもので，機能の拡張，エラーチェック強化，SCSIへの対応，新たなウイルスへの考慮などさまざまな点で改良されています。

前回のDOCTOR.Rでは，SRAMをクリアするzap.xと，DOCTOR.RをSRAMに転送するtfr2.xが別のプログラムになっていましたが，今回はすべての機能をDOCTOR2.X1本にまとめました。また，7月号でお知らせしたように従来のDOCTOR.RはSCSIを採用したX68000 SUPER-HDでは使用できません。もちろん，今回のDOCTOR2.XはSCSIへも対応しています。

さて，DOCTORはチェック用として内部に持つIPLのセルフチェックを行いますが，これがNGとなる場合があります。これはSRAMの一部がビット化けしているためで，機種によってはハードウェア的にSRAMの状態が不安定なケースがあるようです。そこで，今回のDOCTOR2.Xではその場合も想定して，NGとなった際の修復機能も付加しています。ただし，あまり頻繁にNGとなるようでしたら編集部宛にご連絡ください。

なお，SRAMはいくつかのエリアに分かれており，オプションスイッチによって初期化するエリアを指定できます。必要に応じてご使用ください。

ヘルプメッセージ

```
B>doctor2 -?
X68k IPL Checker V1.01 Copyright 1990, Oh!X(中森章)

使用法：doctor [ｵﾌﾟｼｮﾝ]
ｵﾌﾟｼｮﾝ  /t   SRAM への転送を行う
       /b   ブレークキービットだけをねかせる
       /r   リザーブエリア($ED005B～$ED00FF)を初期化する
       /p   プログラムエリア($ED0100～$ED3FFF)を初期化する
       /d   SRAM DISKエリア($ED0400～$ED3FFF)を初期化する
       /a   リザーブエリア以降($ED005B～$ED3FFF)を初期化する
       ﾅｼ    DOCTORを実行する
なお、/t 以外はブレークキービットチェックを行います
```

X68000用
めぞん一刻より暁に鐘は鳴る他2曲

Saitou Akiyoshi
斎藤　彰良

X1/turbo用
涙で綴るパパへの手紙

Sasaki Kouji
佐々木　孝司

あけましておめでとうございます。寒い毎日が続いていますが，皆さんカゼなどひいてませんか？　さて，新年初のLIVE in '91はあの「メゾン一刻」のTVサイズミュージックと，バービーボーイズです。きっとアニメファンにもロックファンにも満足していただけると思います。

響子さぁ～ん，好きじぁ～～～！

新年早々とんでもない小見出しを付けてしまいました。通の人でなくてもわかるかもしれませんね。Live in '91を飾る栄光の1曲目は，めぞん一刻から「暁に鐘は鳴る・93話TVサイズ」です。ちょっと短めの曲ですので，打ち込むのも楽なのではないでしょうか。物足りないと思う人もいるかもしれませんね。そんな人のためにオマケでもう2曲付けてしまいましょう。曲は「響子の悲しみ」と「夜の雨」です。OPMAやOPMDで，バリバリのサンプリング音を駆使した作品が多いなか，異色ともいえる落ち着いた雰囲気をかもし出しています。元気めいっぱいの作品と比べるとちょっと物足りない気もしますが，たまには静かな音楽にひたるのもよいのではないでしょうか。

この作品たちはプリセット音を使っているので，ほかの音楽プログラムなどを演奏させたあとで聴くと音色が変わっているかもしれません。注意してください。

作者の斎藤君にいわせると，これらの作品のなかでは1曲目の「暁に鐘は鳴る」が本命なのだそうです。チャンネル数の不足のために，ハープでコードになるべきところがコードになっていなかったり，いくつかの点で完璧とはいえないそうですが，納得できるデキだとか。「響子の悲しみ」はちょっと悲しい感じのする曲で，第27話の「消えた惣一郎!?」の夜中にかかった曲なんだそうです。「夜の雨」はピアノ曲です。聴いてみると，確かにタイトルの感じがよくわかる気がします。私の勝手な解釈を書いておきましょう。夜の公園の噴水の側にたたずむ2人。水面を静かに叩く雨。黙りこんで気まずくなったふたりをときがやさしく

めぞん一刻

包みこんでくれる……。う～む，考えただけでもロマンチックですねぇ。今月号の発売日はクリスマスのちょうど1週間前ですから，この記事を読んでからイブを迎える人もいることでしょう。"がんばってくださいね（声：島本須美）"。

こちらもキョウコさん・バービーです

さて，今月のX1はMusic BASIC用の「涙で綴るパパへの手紙」，バービーボーイズです。このバンドは男女のツインヴォーカルで構成されています。女の人の名前がキョウコさんなのはただの偶然です。これで男の人の名前がローニンさん（この時期禁句）だったらね……（本当はコンタさんといいます）。

作品はアルバム「Listen」からの選曲です。比較的に古いアルバムで，バービーがバリバリに流行っていたころのものです。私も「Listen」はLP版を持っていたので，楽しみに聴いてみたのですが，どうも聴き覚えがないのです。でも確かに作者の佐々木君のコメントにはそう書いてある。う～む，と考えてみると，CDのほうにはオマケに1曲ついていたのではないだろうかという疑問にぶちあたります。調べてみると確かにありました，CDバージョンの7曲目

バービーボーイズ

ですね。うむ，まいった。

さて，佐々木君は同時に4曲を送ってきてくれました。知名度としては断然上のものである「ごめんなさい」や「泣いたままでlisten to me」などもあったのですが，トータル的にはこの掲載作品がいちばんよくまとまっていたと思います。ただし，全体的に原曲と比べるとテンポが遅く，モタつき感がありますので，テンポを上げてみてはいかがでしょう。具体的には50行の"T145"を"T153"に直してください。これで原曲とほぼ同じになります。聴き比べるとわかると思いますが，モタつき感は減っています。

それと，2小節ごとに遅れる感じがありますので，対策を練りましょう。たとえば，ほかのコマンドにくらべて音色を設定する作業はMMLにとってかなり大変なものになっています。そこで，Iコマンドを同時に4つもやらないで，前の小節の最後の音を若干短めにして，そこにIコマンドを入れてやると，MMLがオーバーワークになる可能性が減ります。MMLは割り込みを禁止して作業をしますので，オーバーワークになるとテンポずれの直接の原因になってしまいます。楽譜どおりに打ち込むよりも，原曲どおりに音が鳴るように心掛けましょう。

（S.K.）

```
  10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  20 /*                                         /*
  30 /*   めぞん一刻 より                  /*
  40 /*                                         /*
  50 /*        「暁に鐘は鳴る」93話TVサイズ /*
  60 /*                                         /*
  70 /*                      By 斎藤 彰良    /*
  80 /*                                         /*
  90 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
 100 m_init()
 110 dim str p(30)[256],in
 120 dim char o(255),v(4,10)
 130 /*
 140 for i=1 to 8
 150   m_alloc(i,4000)
 160   m_assign(i,i)
 170 next
 180 /*
 190 vdata()
 200 pdata()
 210 m_play()
 220 end
 230 /*
 240 /*  MML SET
 250 /*
 260 func t(tt)
 270  r=0
 280   while o(r)<>255
 290   m_trk(tt,p(o(r)))
 300    r=r+1
 310   endwhile
 320  return()
 330 endfunc
 340 /*
 350 /*  VOICE DATA
 360 /*
 370 func vdata()
 380 v={61,15,0,0,0,0,0,0,0,3,0,    /*Violynn
 390 29,0,0,4,0,26,0,8,0,0,0,
 400 12,4,3,4,2, 0,0,8,0,0,0,
 410 12,4,3,4,2, 5,0,6,0,0,0,
 420 12,4,3,5,2, 2,0,4,0,0,0}
 430 m_vset(71,v)
 440 /*
 450 v={0,15,0,0,0,0,0,0,0,3,0,       /*Bass
 460 31,2,0,1,2,42,0,0,6,0,0,
 470 31,2,0,1,2,48,0,0,4,0,0,
 480 31,2,0,1,2,42,0,0,4,0,0,
 490 28,8,2,7,5, 0,0,1,3,0,0}
 500 m_vset(72,v)
 510 /*
 520 v={63,15,0,0,0,0,0,0,0,3,0,      /*Tamb
 530 31,18,7,9,13,0,0, 6,6,1,0,
 540 29,18,7,9,13,0,0, 7,0,2,0,
 550 30,18,7,9,13,0,0,10,7,3,0,
 560 28,18,7,9,13,0,0,13,3,1,0}
 570 m_vset(73,v)
 580 /*
 590 v={ 4,15,0,0,0,0,0,0,0,3,0,      /*Bell
 600 31,15,5,5,5,20,0, 9,7,0,0,
 610 21,10,4,6,3, 0,0, 2,3,0,0,
 620 31, 6,7,6,5,40,0, 4,7,0,0,
 630 21,10,4,6,3, 0,0, 2,3,0,0}
 640 m_vset(74,v)
 650 /*
 660 v={56,15,0,0,0,0,0,0,0,3,0,      /*Guiter
 670 31, 9,0,5,15,39,0,10,5,0,0,
 680 31,10,0,5,15,39,0,8,3,0,0,
 690 29, 8,0,5,15,30,0,0,5,0,0,
 700 31,18,8,8, 0, 0,0,0,3,0,0}
 710 m_vset(75,v)
 720 /*
 730 endfunc
 740 /*
 750 /*
 760 /*  MML DATA
 770 /*
 780 /*
 790 func pdata()
 800 m_tempo(80)
 810 /*
 820 /*  Main
 830 /*
 840 p(0)="@71o4l8p3q7 y48,10 r1r1r1 v10
 850 p(1)="|:c+2&c+ c+ded4.>b&b2 <d2&ddc+>b |1<c+4e2.:||2>a1
 860 p(2)="|:r4ab<c+4&f+4e4.f+c+4c+d|1c+>bbaa4b4<c+1:||2c+>b
baa4f+4b2.ag+&a1&a2.r4<
 870 p(3)="&a2.r4
 880 p(4)="@29 v15o5|:a4.g+16f+16g+4e4f+g+abe4e4<c+4.>b16a16
b4e4f+g+a<c+>|1b4..r16:||2b1&b2v12
 890 p(5)="@71o3l16q8p3 af+g+abg+ab<l8
 900 p(6)="|:3c+2&c+ c+ded4.>b&b2 <d2&ddc+>b |1<c+2e2:||2>a1
:||3a1
 910 o={0,1,2,1,3,4,5,6,255}:t(1)
 920 /*
 930 /*  Main2 & Etc.
 940 /*
 950 p(0)="@74y49,40v6q8p3l16aa<d-d->bb<ee d-d->aabb<d-d- ee
>aabb<dd d-d-ee>aa<dd d-d-ee>aa<dd d-d-ee>aa<dd v7l8@71o4 r16
 960 p(3)="&a4..@61o5v15c2
 970 p(4)="@73v12o5q1|:72c8:| @71 v8 r16
 980 t(2)
 990 /*
1000 /*  Backing(手抜きしてるから間違いだらけ)
1010 /*
1020 p(0)="@74 y50,45 o4 l16 p3 q8 v2 r64
1030 p(1)="aa<d-d->bb<ee d-d->aabb<d-d- ee>aabb<dd d-d-ee>aa
<dd d-d-ee>aa<dd d-d-ee>aa<dd32.
1040 p(2)="v7>|:aa<d-d->bb<ee>:| <d-d->aabb<d-d- ee>aabb<dd
v6 d-d-ee>aa<dd d-d-ee>aa<dd |:d-d-ee>aa<dd:|
1050 p(3)="v5>|:aa<d-d->bb<ee>:| <d-d->aabb<d-d- ee>aabb<dd
v4 d-d-ee>aa<dd d-d-ee>aa<dd |:d-d-ee>aa<dd:|
1060 p(5)="  >  aa<d-d->bb<ee>   <d-d->aabb<d-d- ee>aabb<dd
v4 d-d-ee>aa<dd d-d-ee>aa<dd |:d-d-ee>aa<dd:|
1070 p(6)="v5>aa<d-d->bb<ee v6d-d->aabb<d-d- v8eeaa>aa<dd v7
d-d->aabb<d-d- v5eeaa>aa<dd v3d-d-ee>a v1 a<d v0 d
1080 o={0,1,2,3,3,3,3,3,3,3,3, 5,3,3,6,255}
1090 t(3)
1100 /*
1110 /*  Chord1
1120 /*
1130 p(0)="@114o5p3q8y51,31 r1r1 v6ev7ev8ev9ev10
1140 p(1)="|:eeee f+f+f+f+ ffff |1eeee:||2e8>e8a8b8<c+>b
1150 p(2)="|:aaaa g+g+g+g+ |1f+f+f+f+ <c+c+c+>b:||2f+f+f+f+
bbbr f+f+f+f+ eeer
1160 p(3)="v11o4|:eeee f+f+f+f+ ffff eeee:|eee2
1170 p(4)="l8v12|:4aaaa bbbb aaaa g+g+g+g+:||:8g+:|a4b4
1180 p(5)="v13@75l1o4 |:3c+ddc+:|
1190 o={0,1,2,3,4,5,255}:t(4)
1200 /*
1210 /*  Chord2
1220 /*
1230 p(0)="@114o5p3q8y52,31 r1r1 v6c+v7c+v8c+v9c+v10
1240 p(1)="|:c+c+c+c+ dddd dddd |1c+c+c+c+:||2c+8>r4.ag+
1250 p(2)="|:f+f+f+f+ eeee |1dddd aaag+:||2dddd g+g+g+r dddd
 c+c+c+e
1260 p(3)="v11o4|:c+c+c+c+ dddd dddd c+c+c+c+:|c+c+c+2
1270 p(4)="l8v12|:4f+f+f+f+ g+g+g+g+ f+f+f+f+ eeee:||:8e:|f+
4g+4
1280 p(5)="v13@75l1o4 r64  |:3ef+fe:|
1290 t(5)
1300 /*
1310 /*  Chord3
1320 /*
1330 p(0)="@114o4p3q8y53,31 r1r1 r8 v6av7av8av9av10
1340 p(1)="|:aaaa aaaa aaaa |1aaa@123av11<@125al4>v10 :||2r4
.r2
1350 p(2)="r8|:28 c+:|eerr c+c+c+c+>aa<r8c+
1360 p(3)="v10o3r8|:aaaa aaaa aaaa aaaa:||2aa8a2
1370 p(4)=" |:9r1:|r2
1380 p(5)="v13@75l1o4 r32 |:3abaa:|
1390 t(6)
1400 /*
1410 /*  Harp & Piano4
1420 /*
1430 p(0)="@118o4q8y54,18v11 r1r1r1  r1r1r1 reab<d-ev12ab<v1
3d-1 r1r1r1 @13l16o6q6
1440 p(1)="v14p2|:r4 ag+f+c+>ag+f+ec+>baf+ <<r4 g+f+e>bg+f+e
d>bag+e <<r4 f+ed>af+edc+>ag+f+d
1450 p(2)=" <<|1r4 <c+>baec+>bag+edc+>a:||2 r8o4b<c+df+a<c+>
b2 r4 def+abafd>baf+dc+>a<c+eab<c+ea2
1460 p(3)="o4l8v14 |:rb<c+earr4 >rb<ef+brr4 >rb<efarr4 |1>rb
<c+earr4:||2>reab<c+eab<c+>ebea&ar4
1470 p(4)="|:3r1:|r2 @118o4q8v13 ba-g-ed1 |:4r1:|r2
1480 p(5)="@13p2q6v14 |:o3l16 ab<d-eab<d-eaed->baed->b b<dg-
ab<dg-abag-d>bag-d> b<dg-ab<dg-abag-d>bag-d> |1ab<d-eab<d-eae
d->baed->b:|
1490 p(6)="<<v15ab<d->baed->baed->baed->b
1500 p(7)="o4l8rb<c+earr4 >rb<ef+brr4 >rb<efarr4>a1
1510 o={0,1,2,3,4,5,6,7,255}:t(7)
1520 /*
1530 /*  Bass & Drums
1540 /*
1550 p(0)="y55,10 y3,3
1560 p(1)="@1o3l1 v8 rrr2.e4 |:7a:|a2a4g+4 |:f+ed|1a2a4g+4:|
|2g+2 y2,7 b2dc+2.
1570 p(2)="y3,3@72 v13 o3 l4e|:3y2,2a2a4.e8:|y2,2a2a8<e>e8 |
:4y2,2a2a4.e8:| y2,2a2&y2,7a2
1580 p(3)="l8b4.b<c+4.c+>b4.b<c+8&l16y2,62c+&y2,62c+&y2,63c+
&y2,63c+&y2,63c+&y2,63c+y2,64
1590 p(4)="v13l4d.d8e.e8d.d8e8&l16y2,62e&y2,62e&y2,63e8&y2,6
3e&y2,63e>y2,64
1600 p(5)="l8d4.d e4.e d4.d v14 |:4y2,2e:||:4y2,2d:|v15|:4y2
,18c+:|>y2,18b4<y2,62e16&y2,62e16&y2,64e8
1610 p(6)="|:7>y2,2a4.y2,2ay2,17a4.y2,2e:|l8y2,2a&y2,17a&y2,
17ay2,17al16y2,62a&y2,62a8&y2,63ay2,64<e&y2,64e&y2,64e8l8
1620 p(7)=">|:3y2,2a4.y2,2ay2,17a4.y2,2e:|y2,2a1
1630 o={0,1,2,3,4,3,5,6,7,255}
1640 t(8)
1650 endfunc
```

日本音楽著作権協会　(出)許諾第9072089-001号

リスト2　響子の悲しみ

```
10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
20 /*                                          /*
30 /*    めぞん一刻  より                      /*
40 /*                                          /*
50 /*              「響子の悲しみ」            /*
60 /*                                          /*
70 /*                        By  斎藤  彰良    /*
80 /*                                          /*
90 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
100 dim char ch(4,10)={
110 52,15,0,0,0,0,0,0,0,3,0,
120 13,0,0,6,0,18,0,4,3,0,0,
130  6,9,3,4,5, 1,0,4,7,0,0,
140 13,0,0,6,0, 8,0,8,7,0,0,
150  6,9,3,4,5, 2,0,8,3,0,0}
160 m_vset(69,ch)
170 dim char piano(4,10)={
180 60,15,0,0,0,0,0,0,0,3,0,
190 31, 0,0,0,0,34,0,2,3,0,0,
200 18,11,8,7,3, 0,1,2,3,0,0,
210 31, 8,0,0,2,30,0,6,5,0,0,
220 25,11,9,7,3, 6,1,2,5,0,0}
230 m_vset(70,piano)
240 dim char piano2(4,10)={
250 60,15,0,0,0,0,0,0,0,3,0,
260 31, 0,0,0,0,38,0,2,3,0,0,
270 18,11,8,7,3, 0,1,2,3,0,0,
280 31, 0,0,0,0,40,0,6,5,0,0,
290 31,11,9,7,3, 6,1,2,5,0,0}
300 m_vset(71,piano2)
310 dim str a[256],b[256],c[256],d[256],e[256],f[256],g[256]
320 m_init()
330 for i=1 to 8
340  m_alloc(i,1000)
350  m_assign(i,i)
360 next
370 /*
380 /*   S E T   M M L
390 /*
400 a="t120 @71 l4 q8 o4 v11 y48,00
410 b="|:ggg2<c1> fff2b-1:|
420 c="  ggge- g2<c2>b-1&b-1
430 m_trk(1,a)
440 m_trk(1,b)
450 m_trk(1,c)
460 m_trk(1,b)
470 m_trk(1,c)
480 a="      @71 l4 q8 o4 y49,10 v11 r8
490 b="|:a-f.<e-4.&e-1> ge-.<d.&d1>:|
500 c="a-fcf.f2<d4.&d1&d1>
510 m_trk(2,a)
520 m_trk(2,b)
530 m_trk(2,c)
540 m_trk(2,b)
550 m_trk(2,c)
560 a="      @71 l1 q8 o3 y50,00 v11
570 b="|:g&g f&f:|gg2f2b-1&b-1 @70 o1 l4
580 c="|:a-<<c>b-a->:||:g<b-ag>:|
590 d="f<a-gf>g<e-a-<e-1
600 m_trk(3,a)
610 m_trk(3,b)
620 m_trk(3,c)
630 m_trk(3,c)
640 m_trk(3,d)
650 a="      @71 l1 q8 o3 y51,10 v11
660 b="|:c&c>b-&b-<:|ce-2>b-2<e-1&e-1 @70 o2 l4 r8
670 c="|:e-e-e-e- e-e-e-e- dddd dddd:|cccc>a-<g<c4.>b-1
680 m_trk(4,a)
690 m_trk(4,b)
700 m_trk(4,c)
710 a="      @69 l2 q8 o5 y52,00 v11
720 b="|:rrr4 c. rrr4 >b-.<:|
730 m_trk(5,a)
740 m_trk(5,b)
750 a="      @69 l2 q8 o4 y53,10 v11
760 b="|:rrr4 g. rrr4 f.:|
770 m_trk(6,a)
780 m_trk(6,b)
790 a="      @69 l2 q8 o4 y54,20 v11
800 b="|:rrr4 c. rrr4 >b-.<:|
810 m_trk(7,a)
820 m_trk(7,b)
830 a="      @69 l2 q8 o3 y55,20 v11
840 b="|:rrr4 g. rrr4 f.:|
850 m_trk(8,a)
860 m_trk(8,b)
870 m_play()
```

日本音楽著作権協会　(出)許諾第9072089-001号

リスト3　夜の雨

```
10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
20 /*                                          /*
30 /*    めぞん一刻  より                      /*
40 /*                                          /*
50 /*              「夜の雨」                  /*
60 /*                                          /*
70 /*                        By  斎藤  彰良    /*
80 /*                                          /*
90 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
100 dim char piano(4,10)={
110 60,15,0,0,0,0,0,0,0,3,0,
120 31, 0,0,0,0,34,0,2,3,0,0,
130 18,11,8,7,3, 0,1,2,3,0,0,
140 31, 8,0,0,2,30,0,6,5,0,0,
150 25,11,9,7,3, 6,1,2,5,0,0}
160 m_vset(70,piano)
170 dim char piano2(4,10)={
180 60,15,0,0,0,0,0,0,0,3,0,
190 31, 0,0,0,0,38,0,2,3,0,0,
200 18,11,8,7,3, 0,1,2,3,0,0,
210 31, 0,0,0,0,40,0,6,5,0,0,
220 31,11,9,7,3, 6,1,2,5,0,0}
230 m_vset(71,piano2)
240 dim str a[256],b[256],c[256],d[256],e[256],f[256],g[256]
250 m_init()
260 for i=1 to 8
270  m_alloc(i,1000)
280  m_assign(i,i)
290 next
300 /*
310 /*   S E T   M M L
320 /*
330 a="t148 @71 l4 q8 o5 v11 y48,00
340 b="|:3beg:|b2.>l2|:3aae2.r4e1&e1:|
350 c="|:aaa1:||:bbb1:|  |:aaa1:||:bbb1:|
360 d="|:2 aae1 e1&e1:| aa&a2c ed<ge>
370 e="afaa aaa1 <c1&c1
380 m_trk(1,a)
390 m_trk(1,b)
400 m_trk(1,c)
410 m_trk(1,d)
420 m_trk(1,e)
430 a="      @71 l4 q8 o5 y49,10 v11 r8
440 b="|:9c:|r8r2.>l2ag1g r1r1 ag1ar1r1 ag1gr1r1
450 c="|:ag.<d.>:||:ba.<e.>:| |:ag.<d.>:||:ba.|1<e.>:||2o4r.
460 d="ag1g r1r1 ag1a r1r1 aa&aa>b<gbr
470 e="dagr agr
480 m_trk(2,a)
490 m_trk(2,b)
500 m_trk(2,c)
510 m_trk(2,d)
520 m_trk(2,e)
530 a="      @71 l2 q8 o4 y50,00 v11 r1r1r1
540 b="|:3eee1r1r1:| |:fffr32<f4...>:||:gggr32<g4...>:|
550 c="|:fffr32<f4...>:||:gggr32|1<g4...>:||2o4r4...
560 d="|:2eee1r1r1:| ff&f&fg1&g1 f1f1 eee1r32<e1&e1
570 m_trk(3,a)
580 m_trk(3,b)
590 m_trk(3,c)
600 m_trk(3,d)
610 a="      @71 l2 q8 o4 y51,10 v11 r1r1r1
620 b="r4|:3ee1rr1r1:| |:ffr1:||:ggr1:||:ffr1:||:ggr1:|
630 c="|:2ee1rr1r1:| rc&crr1r1 r1r1 eerr
640 m_trk(4,a)
650 m_trk(4,b)
660 m_trk(4,c)
670 a="      @71 l2 q8 o4 y52,00 v11 r1r1r1
680 b="|:3ccc1c1&c1:| |:dddr16<a4..>:||:eeer16<b4..>:|
690 c="|:dddr16<a4..>:||:eeer16|1<b4..>:||2r4..
700 d="|:2ccc1r1r1:| errrd1&d1 c1c1 ccc1 r16<a1&q1
710 m_trk(5,a)
720 m_trk(5,b)
730 m_trk(5,c)
740 m_trk(5,d)
750 a="      @71 l2 q8 o4 y53,10 v11 r1r1r1
760 b="r4|:3cc1rr1r1:| |:ddr1:||:eer1:||:ddr1:||:eer1:|
770 c="|:2cc1rr1r1:| errrr1r1 r1r1 cc
780 m_trk(6,a)
790 m_trk(6,b)
800 m_trk(6,c)
810 a="      @70 l2 q8 o2 y54,00 v13
820 b="cbg cbr >l2v15|:ca:|ca1&a
830 c="|:3>g<d:|g1 |:3>g<d:|g1 |:>a<f<d1>:|
840 d="|:>b<g<e1>:||:>a<f<d1>:||:>b<g<e1>:|
850 e="|:4ce<ce>:| >a<frr >g<dg1 >a<f>g<d ce<cea1&a1
860 m_trk(7,a)
870 m_trk(7,b)
880 m_trk(7,c)
890 m_trk(7,d)
900 m_trk(7,e)
```


▶来月からアメリカへ出張します。国際電話でパソコン通信やるぞ！　電話代がチョット心配ネ。
原　康之(39)茨城県

```
910 a="      @70 l2 q8 o2 y55,10 v13
920 b="r4gcbgc.>l2v15r4|:g<c>:|grl&r2
930 c="|:3ce:|r1 |:3ce:|r1 |:dar1:|
940 d="|:ebr1:||:dar1:||:ebr1:|
950 e="|:4dg<dg>:| da1&a >b<frr ca>b<gdg<dg
960 m_trk(8,a)
970 m_trk(8,b)
980 m_trk(8,c)
990 m_trk(8,d)
1000 m_trk(8,e)
1010 m_play()
```

リスト4　涙で綴るパパへの手紙

日本音楽著作権協会　(出)許諾第9072089-001号

```
10 ' +----------------------------------------------+
20 ' |                                              |
30 ' |       「 ﾅﾐﾀﾞﾃﾞﾂﾂﾞﾙﾊﾟﾊﾟ 」  By BARBEE BOYS         |
40 ' |                                              |
50 ' +----------------------------------------------+
60 CLS0 : R=0 : FLG=1 : GOSUB 3260
70 TEMPO0 : PLAY"T145" : GOTO 470
80 '== MML PLAY ==
90 '
100 LABEL"!"
110 X$=G$:Y$=H$:IF FLG=1 THEN GOSUB"Drums sub"
120 PLAY    A$;:PLAY":"+B$;:PLAY":"+C$;:PLAY":"+D$;
130 PLAY":"+E$;:PLAY":"+F$;:PLAY":"+G$;:PLAY":"+H$
140 G$=X$:H$=Y$:RETURN
150 '== Drums sub ==
160 LABEL "Drums sub"
170 A=INSTR(H$,"I40") : IF A=0 THEN 200           ' Bass Drum
180 H$=LEFT$(H$,A-1)+"i40o1@v126"+MID$(H$,A+3,LEN(H$)-A)
190 GOTO170
200 A=INSTR(H$,"I38") : IF A=0 THEN 230           ' Snare Drum
210 H$=LEFT$(H$,A-1)+"i38o1@v120"+MID$(H$,A+3,LEN(H$)-A)
220 GOTO 200
230 B=INSTR(G$,"I34") : IF B=0 THEN 260           ' Cymbal
240 G$=LEFT$(G$,B-1)+"i34o4@v127"+MID$(G$,B+3,LEN(G$)-B)
250 GOTO 230
260 B=INSTR(G$,"I36") : IF B=0 THEN 290           ' Close H.H
270 G$=LEFT$(G$,B-1)+"i36o6@v120"+MID$(G$,B+3,LEN(G$)-B)
280 GOTO 260
290 B=INSTR(G$,"I35") : IF B=0 THEN RETURN        ' Open H.H
300 G$=LEFT$(G$,B-1)+"i35o1@v111"+MID$(G$,B+3,LEN(G$)-B)
310 GOTO 290
320 LABEL"P"
330 CALL &H13C
340 POKE &HAB30,&H0,&H40
350 POKE &H461B,0,0,0,0,0,0,0,0,0,0
360 CALL &HAB36
370 END
380 '
390 ' +----------------------------------------------+
400 ' | 11:Guitar         12:Guitar       23:Vocal    |
410 ' | 24:Bass           25:S.Sax        26:S.Sax    |
420 ' | 27:Vocal          34:R.Cymbal     35:Open H.H |
430 ' | 36:Close H.H      38:Snare Drum   40:Bass Drum |
440 ' +----------------------------------------------+
450 '
460 '== Init ==     SPECIAL THANKS TO BALLADE SPORTS CR-X
470 S1$="S2,2,0,12"
480 A$="I25@V12004 L8 Q7 P3  K4  "+S1$   ' ｺﾝﾀ + S.Sax
490 B$="I26@V11804 L8 Q7 P3  K6  "+S1$   ' ｷｮｳｺ + S.Sax
500 C$="I12@V11203 L8  Q7 P3      "       ' ｲﾏｻ
510 D$="I12@V11203 L8  Q7 P3      "       ' ｲﾏｻ
520 E$="                          "       ' ????
530 F$="I24@V12003 L4  Q7 P3      "       ' ｴﾝﾘｹ
540 G$="I36@V12006 L8  Q8 P3      "       ' ｺｲｿ
550 H$="I40@V12001 L4  Q7 P3      "       ' ｺｲｿ
560 "!"
570 A$="=1H4CG+>C+"
580 B$="R@8"+A$+"@16&"
590 C$="":D$="":E$="":F$="":G$="":H$=""
600 "!"
610 '== Intro ==
620 FOR I=1 TO 2
630 A$="O5L8D+E2<G+>C+C2& CR<CG+>C+"
640 B$="O5C+@8"+A$+"@16&"
650 C$="I12O4K6C+C+EC+G+GF+EK5RO5I11S3,2,0,12=1Q0H4E4.=0Q7G+2
660 D$="I12O4K4C+C+EC+G+GF+EK5RO5I11S3,2,0,12=1Q0H4C4.=0Q7E2"
670 F$="O3L4C+C+C+<G+8>C+8 CCC<G+8>C8"
680 G$="I34C4I36C4CCCI35C& I36CCCCI35C4R4"
690 H$="I40CCCC8I38C8 I40CCCI38C"
700 IF R=3 AND I=2 THEN 720
710 IF R<>1 OR I<>2 THEN 760
720    A$="O5D+E2<G+>C+C1& C":B$="O5C+@8"+A$+"@16"
730    F$="L4C+C+C+<G+8>C+8 CCC8C8<G+8>C8"
740    G$="I34C4I36CCCCCI35C& CCI35C4&I36C.R16R4"
750    H$="I40CCCC8I38C8 I40CCC8.I38C16C8I40C8"
760 "!"
770 NEXT
780 IF R=0 THEN R=R+1:GOTO620
790 '== A ==
800 A$="I23@V117L4RO5ED+E C+ED+8C+4."
810 B$=A$
820 C$="I12O4G+G+G+G+G+G+G+G+ EEEEEEE":C1$=C$
830 D$="I12O4C+C+C+C+C+C+C+C+ <AAAAAAA":D1$=D$
840 E$="
850 F$="O3@V123L8C+C+C+C+C+C+C+C+ <AAAAAA>C+D":F1$=F$
860 G$="I34C4I36CCCCI35C4 I36CCCCCCCC"
870 H$=STRING$(2,"I40CI38CI40CI38C")
880 IF R<>2 AND R<>3 THEN 910
890    A$="I23@V117L4RO5ED+E F+ED+8D+4."
900    B$=A$
910 "!"
920 A$="R2R8D+D+8 D+2R2"
930 B$=A$
940 C$="I11F+&G>D+<F+&G>S4,1,0,20=1H3D+4.=0 K6O4F+>F+C+<AG+>G
+D+CK5":C2$=C$
950 D$="I11>F+&GA+F+&GS4,1,0,20=1H3A+4.=0 _3K4O4F+>F+C+<AG+>G
+D+CK5‾3":D2$=D$
960 E$="
970 F$="D+D+D+D+D+<A+4>D+ <F+F+F+G+4G+G+G+":F2$=F$
980 G$="CCCCCCI35CC"
990 H$="I40CI38CI40CI38C I40CI38CI40CC8I38C8"
1000 "!"
1010 A$="RED+E F+ED+8&C+"
1020 B$=A$
1030 C$=C1$:D$=D1$:F$=F1$
1040 G$="I36CCCCCI35C4 I36CCCCCCCC"
1050 H$=STRING$(2,"I40CI38CI40CI38C")
1060 IF R<>3 THEN 1090
1070    A$="RED+E C+ED+8C+4."
1080    B$=A$
1090 "!"
1100 A$="R2R8D+Q0S3,2,0,12=1H4F+ =0Q7G+16R@156"+S1$
1110 B$=A$
1120 C$=C2$:D$=D2$:F$=F2$
1130 G$="CCCCCCI35C4 I36CCCCCCI35C4
1140 "!"
1150 '== B ==
1160 A$="RG+GG+ F+FF+D+"
1170 B$=A$
1180 C$="I12K6O4C+C+C+C+D+D+EF+K5 F+F+F+F+F+F+F+F+":C1$=C$
1190 D$="I12K4O4C+C+C+C+D+D+EF+K5 <BBBBBBBB>":D1$=D$
1200 F$="O3C+C+C+C+C+C+C+_5F+ A@8&B@16BBBBAF+<‾5F+":F1$=F$
1210 G$="I34C4I36CCCCCC CCCCCCCC"
1220 H$=STRING$(2,"I40CI38CI40CI38C")
1230 "!"
1240 A$="O4G+G+>C8D+D+8& F+2R2"
1250 B$=A$
1260 C$="D+D+D+D+D+D+D+D+ EEEERI11E4E"
1270 D$="G+G+G+G+G+G+G+G+ AAAARI11A4A"
1280 F$="G+G+G+G+G+G+G+G+ AAAAR>D4."
1290 G$="CCCCCCI35C4 C4RC4C4."
1300 H$="I40CI38CI40CI38C I40CI38C8R8C8I40CC8"
1310 "!"
1320 A$="RG+GG+ F+FF+D+"
1330 B$=A$
1340 C$=C1$:D$=D1$:F$=F1$
1350 G$="I34C4I36CCCCCC CCCCCCCC"
1360 H$=STRING$(2,"I40CI38CI40CI38C")
1370 "!"
1380 A$="G+GG+8D+D+8& F+2R8E8F+8E8"
1390 B$="G+GG+8D+D+8& F+2R2"
1400 C$="D+D+D+D+D+D+I11D+D+ K6O3A>EA>E4<ABAK5"
1410 D$="G+G+G+G+G+G+I11G+G+ K4O3A>EA>E4<ABAK5"
1420 F$="G+G+G+G+G+G+G+G+ AAB>C+&DC<BA"
1430 G$="CCCCCCCC I35C4I36CCCCI35C4"
1440 "!"
1450 IF R=3 THEN 2240
1460 '== C ==                          ' to$
1470 FOR I=1 TO 2
1480 A$="I23K5@V118L8 O5GGGG4DDD F+F+F+F+4CDE&"
1490 B$="I23K5@V113L8 O5DDDD4<BBB> DDDD4<AB>C&"
1500 C$="O4D4DD4DD4 F+4F+F+4F+F+4"
1510 D$="O3G4GG4GG4 B4BB4BB4"
1520 E$="I27K5@V115L8 O4BBBB4GGG AAAA4F+GA"
1530 F$="O2G4_5>G4‾3D4E4 ‾2<BB>_4F+4‾4D4ED"
1540 G$="I34C4I36CCCCCC CCCCCCCC"
1550 H$=STRING$(2,"I40C4I38C4I40C4I38C8I40C8")
1560 "!"
1570 A$="E4.RR2 R1"
1580 B$="C4.RR2 R1"
1590 C$="A4AA4AA4 A4AA4AA4"
1600 D$="O4F4FF4FF4 D4DD4DD4"
1610 E$="I07FG>C<FG>C<FG F+G>D<F+G>D<GG"
1620 F$="_4E@8&F@16FFE@8&F@16&FFFF‾4 D4C4<B4A4"
1630 G$="CCCCCCCC CCCCCCI35C4"
1640 H$=STRING$(2,"I40CI38CI40CI38C")
1650 IF I<>2 THEN 1730
1660    E$="I07FG>C<FG>C<FG >ADCGC<G>F+C"
```

▶ボーナスが出るからハードディスクを買います。80Mバイトにしようか，40Mバイトにしようか……と，考えているると仕事が手につきません。　鶴田　啓介(21)福岡県

```
1670     F$="_4E@8&F@16FFE@8&F@16&FFFF¯4 D4C4<B4AB"
1680     IF R<>2 THEN 1730
1690        A$=S1$+"E4.RR2 R2RI25@V120 =1H4O4CG+>C+"
1700        B$=S1$+"C4.RR2 R2RI26@V118 =1H4O4R@8CG+>C+@16&"
1710        E$="I07FG>C<FG>C<FG >ADCGC<G>EF+"
1720        F$="_4E@8&F@16FFE@8&F@16&FFFF¯4 D4C4<B4G+B"
1730 "!"
1740 NEXT I
1750 IF R=1 THEN R=R+1:GOTO800
1760 '== D ==
1770 A$="D+E2<G+>C+C2& CR<CG+>C+"
1780 B$="C+@8"+A$+"@16&"
1790 C$="I12K6C+C+EC+G+GF+EK5 RO5I11S3,2,0,12=1Q0H4E4.=0Q7G+2
"
1800 D$="I12K4C+C+EC+G+GF+EK5 RO5I11S3,2,0,12=1Q0H4C4.=0Q7E2"
1810 E$="
1820 F$="O3L4C+C+C+<G+8>C+8 CCC<G+8>C8"
1830 G$="I34C4I36Y190,13CCCCCC CCCCCCCC"
1840 H$=STRING$(2,"I40CI38CI40CI38C")
1850 "!"
1860 A$="D+E2<G+>C+C2& CRF+C+F+"
1870 B$="C+@8"+A$+"@16&"
1880 F$="C+C+C+<G+8>C+8 CCCC8C+8"
1890 G$="CCCCCCCC CCCCCCR4"
1900 H$="I40CI38CI40CI38C I40CI38CI40C8I38C8I40C8I38C8"
1910 "!"
1920 A$="G+A2C+F+F2& FR<CG+>C+"
1930 B$="F+@8"+A$+"@16&"
1940 C$="I12O4K6F+F+AF+>C+C<BAK5 O5RI11=1Q0F4.=0Q7A2"
1950 D$="I12O4K6F+F+AF+>C+C<BAK5 O5RI11=1Q0C+4.=0Q7F2"
1960 F$="_4F+F+F+C+8F+8 FFFF8F+8¯4"
1970 G$="I34C4I36Y190,13CCCCCC CCCCCCCC"
1980 H$=STRING$(2,"I40CI38CI40CI38C")
1990 "!"
2000 A$="D+E2<G+>C+C2.& C"
2010 B$="C+@8"+A$+"@16"
2020 C$="I12O4K6C+C+EC+G+GF+EK5 O5RI11=1Q0E4.=0Q7S4,1,0,14=1G
+2=0"
2030 D$="I12O4K4C+C+EC+G+GF+EK5 O5RI11=1Q0C4.=0Q7S4,1,0,14=1E
2=0"
2040 F$="C+C+C+<G+8>C+8 CCCC8C+8"
2050 "!"
2060 A$="S2,1,0,20=1H1G+4<G+4>G+4<G+4 >G+4<G+2&=0G+G+@8&A@8&B
@8"
2070 B$=A$
2080 C$="I12O4"+STRING$(16,"D+")
2090 D$="I12O3"+STRING$(16,"G+")
2100 F$="O2L8G+G+_4>G+<¯4G+G+G+EG+ G+G+_4>G+<¯4G+_4>G+<¯4G+EG
+"
2110 G$="I34C4I36Y190,13CCCCCC C4R2."
2120 H$="I40CI38CI40CI38C I40CI38C8.C16C8G8C4"
2130 "!"
2140 A$="O5C+4<B4A4G+4 F+4E4D+4G+>C+"
2150 B$="R@16"+A$+"@8"
2160 C$="I11C+4<B4A4G+4 >F+4E4D+4D4"
2170 D$=C$
2180 F$="O3C+4<B4A4G+4 >_3F+4E4¯3D+4D4"
2190 G$="I34C4C4C4C4 C4C4R4C4"
2200 H$="I40CCCC CCC8C8I38C4"
2210 "!"
2220 R=R+1:GOTO 620
2230 '== E ==
2240 FOR I=1 TO 4
2250 A$="I23K5@V118L8 O5GGGG4DDD F+F+F+F+4CDE&"
2260 B$="I23K5@V113L8 O5DDDD4<BBB> DDDD4<AB>C&"
2270 C$="O4D4DD4DD4 F+4F+F+4F+F+4"
2280 D$="O3G4GG4GG4 B4BB4BB4"
2290 E$="I27K5@V115L8 O4BBBB4GGG AAAA4F+GA"
2300 F$="O2G4_5>G4¯3D4E4 ¯2<BB>_4F+4¯4D4ED"
2310 G$="I34C4I36CCCCCC CCCCCCCC"
2320 H$=STRING$(2,"I40C4I38C4I40C4I38C8I40C8")
2330 "!"
2340 A$="E4.RR2 R1"
2350 B$="C4.RR2 R1"
2360 C$="A4AA4AA4 A4AA4AA4"
2370 D$="O4F4FF4FF4 D4DD4DD4"
2380 E$="I07FG>C<FG>C<FG F+G>D<F+G>D<GG"
2390 F$="_4E@8&F@16FFE@8&F@16&FFFF¯4 D4C4<B4A4"
2400 G$="CCCCCCI35CC I36CCCCCCI35C4"
2410 H$=STRING$(2,"I40CI38CI40CI38C")
2420 IF I<>4 THEN 2460
2430     E$="I07FG>C<FG>C<FG >ADCGC<G>F+C"
2440     F$="_4E@8&F@16FFE@8&F@16&FFFF¯4 D4C4<B4AB"
2450     G$="CCCCCCI35CC RCRCC4C4"
2460 "!"
2470 NEXT I
2480 '== F ==
2490 A$=S1$+"I25@V120O5 =1H4RDGF+GDE<B >D4.C4D4E"
2500 B$=S1$+"I26@V118O5 =1H4R@16RDGF+GDE<B >D4.C4D4E@8"
2510 C$="I11@V115O4 D4DD4DD4 F+4F+F+4F+F+4"
2520 D$="I11@V115O3 G4GG4GG4 B4BB4BB4"
2530 E$="I07@V120O4 RG16B16>DDEE<BB RF+16B16>DDEE<BB"
2540 F$="O2L8GG_4>G4¯4D4E4 <BB>_4F4¯4D4ED"
2550 G$="I34C4I36Y190,13CCCCCC CCCCCCCC"
2560 H$=STRING$(2,"I40CI38CI40CI38C")
2570 "!"
2580 A$=STRING$(8,"E@8&F@16")
2590 B$=A$
2600 C$="A4AA4AA4"
2610 D$="F4FF4FF4"
2620 E$="FG>C<FG>C<FG"
2630 F$="E@8&F@16FFE@8&F@16&FFFF"
2640 G$="CCCCCCCC"
2650 H$="I40CI38CI40CI38C"
2660 "!"
2670 IF R=4 THEN 2780
2680 A$="D2.R4"
2690 B$=A$
2700 C$="A4AA4AA4"
2710 D$="D4DD4DD4"
2720 E$="F+G>DEDE<GG"
2730 F$="D4C4<B4A4"
2740 G$="CCCCCCCR"
2750 H$="I40CI38CI40C8I38C8R8C8"
2760 "!"
2770 R=R+1:GOTO 2490
2780 A$="D2R4O4D@8E@8F@8<B-"
2790 B$=A$
2800 C$="A4AA4AA4"
2810 D$="D4DD4DD4"
2820 E$="O5A4.G2&G8"
2830 F$="D4C4<B8A8R>F+8"
2840 G$="CCCCCCC4"
2850 H$="I40CI38CI40CI38C"
2860 "!"
2870 '== G ==
2880 A$="O4G4G4GF+FE-2.& E-RR":A1$=A$
2890 B$="R@16"+A$+"@8"
2900 C$="@V107"+STRING$(4,"I12O4BBI11BB")
2910 D$="@V107"+STRING$(4,"I12O4GGI11GG")
2920 E$="@V107"+STRING$(4,"I12O4E-E-I11E-E-")
2930 F$="O3G4G4GF+FE-4 G4>E-<G>E-<G":F1$=F$
2940 G$="I34C4I36Y190,13CCCCCC CCCCI34C4I36Y190,13CC"
2950 H$="I40CI38CI40CC8I38C8 I40CI38CI40C8C8I38C"
2960 "!"
2970 A$="O4G4G4GAB>C+2.& C+RR"
2980 B$="R@16"+A$+"@8"
2990 F$="O3G4G4GAB>C+4<C+4>C+4<C+>C+4"
3000 "!"
3010 A$=A1$
3020 B$="R@16"+A$+"@8"
3030 F$=F1$
3040 "!"
3050 A$="O4G4G4GAB>C+& T135L1Q8C+"
3060 B$="R@16"+A$+"@176"
3070 C$="I12BBI11BBI12BBI11BB& B1&"
3080 D$="I12GGI11GGI12GGI11GG& G1&"
3090 E$="I12E-E-I11E-E-I12E-E-I11E-E-& E-1&"
3100 F$="O3G4G4GAB>C+8& C+8<C+&C+2.&"
3110 G$="I34C4I36Y190,13C.R16R4.I34C& C1&"
3120 H$="I40CC8.I38C16C8G8C8I40C8& C1&"
3130 "!"
3140 A$="L16C+F+BA+A+BA+G+A+ED+DC+1& C+"
3150 B$="R@8"+A$
3160 C$="B1& B1"
3170 D$="G1& G1"
3180 E$="E-1& E-1"
3190 F$="C+1& C+1"
3200 G$="I34C1& C1"
3210 H$="I40C1"
3220 "!"
3230 KEY3,CHR$(34)+"P"+CHR$(34,13)
3240 END
3250 '== TONE DATA ==
3260 MEM$(&HB2F8,36)=HEXCHR$("D1 50 32 20 40 40 19 1F 11 0F 1
F 1F 1F 1C 05 08 08 06 04 04 04 04 34 23 54 56 00 00 00 00 00
 D2 87 00 02 00 ")' 11:Guitar
3270 MEM$(&HB31C,36)=HEXCHR$("D1 50 32 20 40 40 15 1F 11 0F 5
F 5F 5F 5C 09 09 09 0D 04 04 04 06 66 66 66 A6 00 00 00 00 00
 D2 87 00 02 00 ")' 12:Guitar
3280 MEM$(&HB4A8,36)=HEXCHR$("FA 00 60 21 60 71 1E 14 24 00 1
2 14 14 12 04 04 04 04 04 04 04 04 06 06 06 06 00 00 00 00 00
 00 80 00 00 00 ")' 23:KONTA
3290 MEM$(&HB4CC,36)=HEXCHR$("DC 00 40 40 70 70 25 11 16 00 1
F 1F 1F 1F 04 04 04 09 00 00 02 05 55 05 55 55 00 00 00 00 00
 00 80 00 00 00 ")' 24:Bass
3300 MEM$(&HB4F0,36)=HEXCHR$("FA 00 31 22 41 61 1E 32 28 0D 1
F 4F 0F 14 05 0A 0A 0A 0A 01 00 00 97 A7 17 07 00 00 00 00 00
 00 80 00 00 00 ")' 25:SopranoSax
3310 MEM$(&HB514,36)=HEXCHR$("F2 00 31 21 41 61 0E 28 28 0D 1
C 0F 0F 0F 00 08 08 0A 00 01 00 00 05 55 15 07 00 00 00 00 00
 00 80 00 00 00 ")' 26:SopranoSax
3320 MEM$(&HB538,36)=HEXCHR$("FA 00 61 21 61 71 1E 14 24 00 1
2 12 12 12 04 04 04 04 04 04 04 04 05 05 05 05 00 00 00 00 00
 00 80 00 00 00 ")' 27:KYOKO
3330 MEM$(&HB634,36)=HEXCHR$("FB 00 70 06 09 31 08 19 11 00 1
E 1E 1F 1C 04 00 00 0C 00 00 80 00 F2 F1 F4 F6 80 00 00 00 0C
 C8 80 00 02 00 ")' 34:R.Cymbal
3340 MEM$(&HB658,36)=HEXCHR$("FC 00 0F 01 0A 0F 00 11 00 17 5
E 9F 5E 9F 05 05 07 0A C0 46 C2 86 02 57 12 57 00 00 00 00 00
 00 80 00 00 00 ")' 35:Open H.H
3350 MEM$(&HB67C,36)=HEXCHR$("FB 00 0E 06 07 00 0F 1B 11 05 1
A 1A 1A 16 04 08 16 92 40 40 80 00 32 72 BA F8 00 00 00 00 00
 00 80 00 00 00 ")' 36:CloseH.H
3360 MEM$(&HB6C4,36)=HEXCHR$("FC 00 7C 01 70 01 04 00 07 00 1
F 1C 1E 1F 00 0F 0F 0A C0 00 00 00 05 A5 A5 A5 00 00 00 00 F0
 C8 80 00 02 00 ")' 38:Snare Dr.
3370 MEM$(&HB70C,36)=HEXCHR$("F8 00 01 0E 00 50 00 00 07 00 1
E 1E 19 1D 1A 1C 10 07 40 C0 40 00 FD FE F8 F8 00 00 00 00 D0
 C8 80 00 00 80 ")' 40:Bass Drum
3380 RETURN
```

電話番号が変わります

Ogikubo Kei 荻窪 圭

平成3年1月1日，あなたの電話番号が変わります。東京都以外の人にはあまり関係ないけど，当事者たちにとっては大問題。だって，電話帳やらなんやらを変更しないといけませんからね。というわけで，簡単ながらも便利なプログラムの登場です。

今回はいきなり番外編である。まあ，こんなこともあろう。

さて，NTTという企業がある。ダイヤルQ^2で儲けているNTTだ。Q^2ってのはたいていエッチなことに使われている。NTTさんは企業とかがビジネスで使う用途を想定したそうだが，眉唾である。伝言ダイヤルを見ていれば，Q^2がどう使われるかなんて明らかではないか。これはもう確信犯である。そんでもって，伝言ダイヤルに飽きたらなくなった人はダイヤルQ^2を始め，電話代が30万円とか100万円とかになったりするのだ。若者の心の隙間から生き血をすすりとるNTTと呼んであげよう。さすが，民間企業になって営利を追求するようになるとやることが違う。JRも見習ってスケベ列車でも走らせる（もちろん表向きはビジネス利用ということにするのだ）といいかもしれない。

さて，そのNTTさんのいうことには，平成3年，つまり1991年という21世紀の10年前という記念すべき年に入ったとたん，東京03局の市内局番を4桁にしようというのだ。元旦の午前2時にである（どうして2時かは知らないけれど）。これまでも，3桁では足りない，ってことで新しい加入者をどんどん5で始まる4桁の市内局番にしていたのだが(Oh!X編集部も4桁)，それでもまだ足りないらしく，古い人も一斉に4桁にしてしまう計画らしい。どうして古い人まで4桁にする必要があるかというと，177-2533ってかけてもちゃんと天気予報にかかるようなもの，ってとこから類推してください。細かくは割愛。

東京の人口はそんなに増えなくても，電話の加入者は増えるのだ。会社の数は増えるし，会社は回線を増やすし，FAX専用回線なんて引いたりするからだ。そういうわけで，番外編は時事ネタの局番変更。「大人のためのX68000」には格好のネタではないか。

そこで考えた。局番変更に向けて，どうするのが一番楽にすむだろうか。Kamikazeのマクロでできるか。CARD PRO-68Kではどうか。手作業はいやだし。うーん。

結論。あまりに安易なところへ落ち着いてしまってもうしわけないが，X-BASICを使うことにした。X-BASICなら，

1) 誰もが持っている
2) テキストファイルならどんな形式でもOK

というメリットがあるからである。

プログラミングへの道

出来上がったのは簡単なファイル入出力のプログラムである。入力ファイルひとつに出力ファイルひとつという一番単純なパターンだ。入力ファイルから1レコード読んでは加工して1レコード書き出す，ってのをファイルがなくなるまで繰り返すだけだ。1レコードってのはまあ，1行のこと。Human68kでいう1行ってのは(MS-DOSでも同じだが)，CR＋LFを区切りとした1文字列だ。

ちなみに，CRってのはキャリッジリターンのことで，日本語でいうと“復帰”である。どこに“復帰”するかというと，行の先頭だ。LFってのはラインフィードのことで，日本語では“行送り”，または“改行”である。だからCR＋LFを復帰改行という。2つあわさって初めて一般でいう“改行”が成り立つのだな。タイプライター文化の名残といっていいだろう。

では，もちっと細かいアルゴリズムを考える。

まず最初にNTTのパンフレットを見る。“現在3ケタの市内局番の前に「3」をつけ，4ケタの市内局番になります”

これだけ。簡単そう。

頭の中で整理する。“「03-」があったら市内局番を取り出して，3桁だったら頭に「3」を付加する”だけでいい。

これをプログラムできるところまで分解する必要がある。

1) 入力ファイルの形式を決める。

これはテキストファイルならなんでもOKとしよう。

2) 出力ファイルの形式を決める。

これは入力ファイルに準ずる。

問題は，“3”を付加したためにその行が1桁増えてしまうという現実をどうするか，ってことだ。私はご存じのとおり，怠慢である。そこで「気にしない」ことにした。CSV形式なら，項目長を無視できるので気にしなくてもいい。区切りなしASCIIだと困るが，区切りをタブコードにしているなら，きっと，問題ない。区切りありASCII形式なら項目長を無視できるので，問題ない。ということで，問題はないのだ。

3) アルゴリズムを決める。

まず，「03-」を探す。これだと住所の項目や郵便番号の項目に「103-55」なんてのがあったり，電話番号が「052-403-9999」なんてのがあったりしてもマッチングしてしまうが，これも気にしない。次のチェックがあるからだ。

「03-」の次の4桁を取り出す。この4桁目が「-」かどうかで市内局番が3桁か4桁か判断できるから。郵便番号や市内局番に「03-」が含まれていても，その4桁後に「-」がくることは（おそらく）ない。住所に含まれているときも，『X03-XXX-XX』などというケースはきっと稀だ。

ここで，この連載の精神が明らかになる。つまり，「たった1回しか使わないプログラムだから，簡潔であるべし。余計な気遣い

は無用」。

今年から来年にかけてしか使わないものに，手間暇かけるのは無駄というものだ。予定外のマッチングなんて滅多にないのだから，そういうケースはあとから手作業で直したほうがずっと早い。少なくとも，そういった稀なケースのための処理をプログラムに追加するよりは，だ。

第1回「大人のためのX-BASIC入門」(?)

そんなこんなでリストのプログラムが出来上がった。縁起ものなので説明はする。

変更するデータファイルであるが，ファイル名を入力させるのも間抜けなので，プログラム中に書いてしまった。変数宣言部のFN（File Nameの略ね）に書いてある。ここを自分に都合のいいものに変更する。

続いて，NFNには拡張子をOLDにしたファイル名が入る。

ちょっとややこしいが，“TEL.DAT”てなファイルをこのプログラムにかけると，“TEL.DAT”を“TEL.OLD”にリネームし，“TEL.OLD”を入力ファイルに，“TEL.DAT”を出力ファイルにする。こうすると，見かけ上，TEL.DATが書き換えられて，バックアップファイルとしてTEL.OLDが作られたように見えるのだ。

入力されたファイルは1行ずつ“R-BUF”（リードバッファの略ね）という変数に読み込まれる。R-BUFはstr型で宣言しているので，MAXでも255文字である。これ以上長い行には対応していない。これはひとえに私の怠慢であるが，そのあたりを柔軟にすると，プログラムが長くなるのでやめた。だから，データファイルにはCSV形式など1行がコンパクトにおさまる形式を勧める。

読み込まれたらまず，“03-”があるかどうかチェックし，なければその行を書き出す。write時，行の末尾にCR+LFを付加すること。

とりあえず，“03-”を見つけたら次の4桁をチェック。必要なら変更して，そこまでを書き込み用のバッファである“W-BUF”（ライトバッファの略ね）へ追加する。そして，残りの部分からまた“03-”の検索を開始する。

元のファイルから1行ずつR-BUFへ読み込んで，必要なら加工しながら，W-BUFへ代入し，1行分終わったらファイルへ書き出すという作業をレコードがなくなるまで続けるわけだ。

ただこれだけだが，repeat～untilでなく，while～endwhileを使いたいという理由により，レコードのreadが2カ所にある。whileの判定にEOFかどうか（End Of File，つまり，ファイルが終わりかどうかをチェックすること。freads関数のリターンコードでわかる）を使っているため，whileの前に一度readしておく必要があるのだ。

まあ，こういう書き方もあるのだと思っていただければいい。私は諸般の事情により，こういう考えに慣れてしまったのだ。

全レコードの処理を終えたら，クローズして終了する。

不親切な本プログラムの使い方

なにはさておき，更新したいデータファイルを用意する。Kamikazeならカルクシートから標準テキスト形式で保存すれば得られる。CARD PRO-68KやCYBER NOTE PRO-68Kならコンバート機能でCSV形式にコンバートする。電話番号は“XXX-XXX-XXXX”のように，ハイフン『-』で局番を区切ってあること。“XXX(XXX)XXXX”とか“(XXX)XXX-XXXX”ってな書き方には対応していない。どうしても括弧を使いたい，って人は，プログラムを書き換えてもらいたい。簡単だから。もちろん，『03-』を省略してある怠慢なものにも対応していない。

用意できたら，エディタ(X-BASICで打ち込んでも構いませんが）でこのリストを打ち込む。ファイル名はちゃんと自分の使うデータファイル名に直すこと。

そんでもって，セーブし，BASICからloadコマンドでロードする。

ここで実行するわけだが，注意すべきは“このプログラムは必要最小限の処理しかしない”ということである。ファイル名を間違えた，とか，1行あたり500バイトもあった，なんてケースには対応していない。たとえば，リネームする関数はリネームする予定の名前と同じ名前のファイルがあるとそこで止まってしまうが，だからといってこのプログラムは何もしないのだ。そういったエラーが出て止まったら，自分で対処してほしい。

冷たいが，エラー処理のためにプログラムが長くなるのとどちらがいいか，っていわれると，私は「たった1度や2度しか使わないプログラムで100行程度なら，短い方がいい」と答えるのだ。長くなるとそれだけ打ち間違いも増えるわけだし。

そういうわけで，最初は，4，5行の短いテスト用データファイルを用意して実験することを勧める。

うまくいかなかったときでも，拡張子がOLDのファイルにちゃんと元のデータが残っているので，こいつを元のファイル名に戻してやり直せば大丈夫だ。

そういったわけで，Cにコンバートしてコンパイルするのは勧めない。止めたほうがいい。動く保証もしない。

なお，こういういい加減なプログラムでファイル処理を行うのは本来危険なので，データファイルのバックアップは取ってお

リスト

```
 10 /*************************************************************/
 20 /*       1991年からの03局局番変更対応プログラム              */
 30 /*                              90/11/09  K.Ogikubo          */
 40 /*************************************************************/
 50 /*************************************************************/
 60 /*               変数・定数定義部                            */
 70 /*************************************************************/
 80 str FN[64]="D:TEL.DAT"              /* DB file name       */
 90 str NFN[64]                         /* 拡張子がoldのもの  */
100 str R_BUF[255]                      /* Read Buffer        */
110 str W_BUF[255]                      /* Write Buffer       */
120 str TOKIO[4]                        /* 市内局番           */
130 str FIRST[255]                      /* ポインターより前   */
140 str LAST[255]                       /* ポインターより後ろ */
150 str CRLF[2]                         /* CR+LF              */
160 int RC=0                            /* 汎用リターンコード */
170 int FRC                             /* Fileリターンコード */
180 int R_PTR                           /* Record Pointer     */
```

こう。

テスト用のファイルでうまくいったら，平成3年の元旦の午前2時を待つ。そして，午前2時になったら「えいやっ」と実行する。うまくいけば新しいアドレス帳で新年を迎えることができるわけだ。

完了したら，また，元の形式(KamikazeなりCARD PRO-68Kなり）に戻して，何事もなかったかのようにその住所録を使えばいい。それだけである。

これにて「大人のためのX68000」番外編はおしまい。次から「大人のためのX-BASIC入門」が始まる，てなことは（きっと）ないからご安心を。

BASIC的トラブルシューティング

打ち間違いやらなにやらで，うまくいかないことは多い。そういうときはBASICのメリットをフルに生かしてごまかそう。

1) 実行途中で止めてしまったり，エラーで止まったりした場合

ファイルがopenしたままになっていると思われるので，

FIN()

をまず実行しよう。FIN()ってのはファイルをクローズする部分（終了処理）の関数名だ。BASICはこうして特定の関数だけを実行できるので，便利である。

そして，デバッグする。

2) 同じファイル名があるのでリネームできない，っていわれた場合

よくあるケース。こいつも，

kill NFN

でいい。これは元のファイル名にデータがあって，拡張子OLDのものがじゃまな場合だ。エラーで止まったときは変数の内容は保持されているので，NFNには拡張子がOLDのファイル名が入っている。

拡張子.OLDのものを元の名前に戻したいときは，

kill FN

で元のファイル名を消して，

frename(NFN, FN)

を実行すればいい。BASICの利点だ。

動作確認は私が使っている電話帳ファイルによって行っているが，本文で書いたようにうまくいかないケースは多々ありうる。そういうときはごめんなさい。

```
190 int RB_LEN                                /* Read Buffer Length */
200 int COUNTER=0
210 CRLF=chr$(13)+chr$(10)                    /* CR+LF              */
220 /*******************************************************************/
230 /*        メインプログラム                                         */
240 /*******************************************************************/
250    INIT()                                  /* 初期処理            */
260    while not(FRC)
270       R_PTR=instr(1,R_BUF,"03-")           /* 03-を探せ           */
280       if R_PTR=0 then {
290          W_BUF=R_BUF                       /* 03-はなかった処理   */
300       } else {
310          while R_PTR
320             TOKIO=mid$(R_BUF,R_PTR+3,4)
330                                   /* 市内局番の取り出し           */
340             if right$(TOKIO,1)="-" then {
350                FIRST=left$(R_BUF,R_PTR+2)
360                LAST=right$(R_BUF,RB_LEN-(R_PTR+6))
370                W_BUF=W_BUF+FIRST+"3"+TOKIO
380                                   /* 市内局番に3を追加            */
390                print TOKIO;" を 3"+TOKIO;" に変更"
400                COUNTER=COUNTER+1
410             } else {
420                FIRST=left$(R_BUF,R_PTR+6)
430                LAST=right$(R_BUF,RB_LEN-(R_PTR+6))
440                W_BUF=W_BUF+FIRST
450                             /* そのまま新しいバッファへ           */
460             }
470             R_BUF=LAST                      /* 旧バッファ更新     */
480             RB_LEN=strlen(R_BUF)
490             R_PTR=instr(1,R_BUF,"03-")
500          endwhile
510          W_BUF=W_BUF+R_BUF
520       }
530       RC=fwrites(W_BUF+CRLF,FNO2)
540       FRC=freads(R_BUF,FNO1)
550       W_BUF=""
560       RB_LEN=FRC
570    endwhile
580    FIN()                                   /* 終了処理            */
590 end
600 /*******************************************************************/
610 /*        初期処理(ファイル名変更、ファイルオープンなど)           */
620 /*******************************************************************/
630 func INIT()
640    int SRC
650    FNCHG()                            /* ファイル名.oldの作成 */
660    FNO1=fopen(NFN,"r")
670    FNO2=fopen(FN,"c")
680    FRC=freads(R_BUF,FNO1)             /* 第1read              */
690    W_BUF=""
700    RB_LEN=FRC
710 endfunc
720 /*******************************************************************/
730 /*     ファイル名変更処理                                          */
740 /*******************************************************************/
750 func FNCHG()
760    int RC
770    str PERIOD="."
780    RC=instr(1,FN,PERIOD)
790    if RC=0 then {
800       RC=strlen(FN)+1
810       FN=FN+PERIOD
820    }
830    NFN=left$(FN,RC)+"OLD"
840    print FN+" を "+NFN+" にリネームします。"
850    frename(FN,NFN)
860 endfunc
870 /*******************************************************************/
880 /*        終了処理                                                 */
890 /*******************************************************************/
900 func FIN()
910    print "修正した電話番号は ";COUNTER;" 個です。"
920    fclose(FNO1)
930    fclose(FNO2)
940 endfunc
```

宇宙要塞CADの逆襲 その2

プロジェクトチームDōGA

MAX田口

前回に引き続き，紙飛行機2号の作成です。分量が多かったので2回に分かれてしまいましたが，今回でいよいよ完成です。前回のおさらいはちゃんとできていますか？ 覚えていない方は，1990年11月号を手元に置いて読んでください。

はじめに

皆さん，宇宙要塞CADは我々（私？）の想像以上に強固だった。残念ながら，11月号だけでは，とても陥落させることができなかったのだ。どうしても分量が多くなってしまって，2回に分けるしかなかったわけだ。おかげで，今月も原稿を書くはめになった。これは，Oh!X編集部の陰謀か，はたまた極悪かまたのさしがねか，それとも私が悪いのか？

しかし，こうなったらじたばたせずにさっさとCADでもなんでも攻略して，ナイアスでもクリアしようっと。

前回のおさらい

それじゃあ，さっそくCADを立ち上げて，前回作った「紙飛行機2号」をロードしよう。前回は，ページの都合上セーブの仕方について書けなかったけど，ちゃんとセーブしておいてくれたかな。もしセーブしないでそのまま終了した人がいたら，その人は，残念だけどもう一度最初から作り直すしかない（まぁ，そんな人はいないと思うけど……）。ロードの仕方は，FileModeのloadのところをクリックして，ファイル名を入力するだけだ。ファイルのロードが終わると画面に，何か表示されるけど，今回は何もせずに，「固定終了」を選択する。これだけで，ファイルのロードは終わりだ。

さて，ロードが終わったからといって，いきなり「Edit Mode」に入った人は，いないかな。CADを立ち上げたら，真っ先にするのは，面を作ることじゃなくて，そう，Cパネルの設定変更だ。今回は2度目だから変更する点だけを書いておくから，変更の仕方を忘れた人は，11月号を見るように。PANEL 1の変更点は，表示スケールを「8」にすること。gridを「200」にすること。アトリビュートの指定を「body」にすること，の3点だ。そして，次にPANEL 4の3Dカーソルの移動幅を「100」にする。これが済んだら，再びCパネルの表示をPANEL 1にして，設定は終わりだ。

さて，アトリビュートの指定をするところで気がついた人もいるかもしれないけど，前回たくさん指定したアトリビュートが，「body」だけになっている。なぜかというと，前回では，アトリビュート名が「body」の面しか作らなかったので，前回登録したほかのアトリビュート名が消えてしまった（というかセーブされなかった）のだ。そこで，残りのアトリビュート名を再び入力しないといけない。

アトリビュート名の登録は，Attribute ModeのAttri.登録で行うんだけど，覚えていたかな。Attri.登録をクリックしたら，今回の「紙飛行機2号」で使う「body」以外の4つのアトリビュート名を順番に入れていこう。「cockpit」，「engine」，「wing」，「missile」と，4つのアトリビュート名をキーボードから順に入力し終わったら，ESCキーを押してアトリビュート登録を終了しよう。

これで，前回セーブしたときと同じ状態になったので，安心して制作の続きができるぞ。

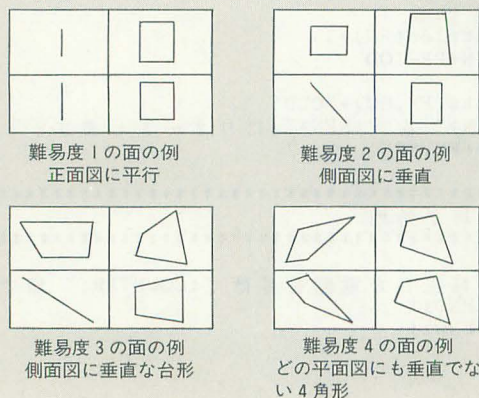
図1
難易度別の面の例

胴体部分の作成の続き

<その4>難易度3の面の作り方

胴体部分は，11月号でほとんど完成している。残っているのは，機首と胴体を結ぶくびれた部分だけだ。これは難易度3の面を使って作る。難易度3の面といっても，作り方はほとんど難易度2の面と変わらない。

さっそく，Edit Modeに入って，アトリビュート名が「body」になっているのを確認したら，面を作り始めよう。さっきもいったとおり，いまから作るのは難易度3の面といっても簡単なほうなので，全然難しくない。だから，前回の復習だと思って作ってもらえばいい。

今度作る面は，側面図を主に使って作る。まず3Dカーソルを，（−400,200,100）のところに持っていく。この点が第1頂点になるので，スペースキーを押して，点を確定する。第1頂点を決定したら，次に3Dカーソル

を下に1と2分の1マス分（−300）動かす（座標は（−400,200,−200）になる）。ここが第2頂点になる。今度は，3Dカーソルを，右に1マス分（+200），上に2分の1マス分（+100）動かす。そして，今度はマウスカーソルを正面図のところに持っていって，3Dカーソルを左に2分の1マス分（−100）動かす（ちゃんと座標が（−200,100,−100）になったかな）。ここが第3頂点になる。あとは，マウスカーソルを側面図に戻して，3Dカーソルを上に1マス分（+200）だけ動かして（−200,100,100）にしたあと，第4頂点を確定して，面を確定すればいい。これで，胴体のくびれた部分の右側の面は完成だ。ほら，簡単だったろ？ これで，面の作り方を思い出したかな？

あとは同じようにして，残り2つの面を作ればいいわけだ。

＊

・四角形　第1頂点（−200, −100, −100）
　　　　　第2頂点（−200, −100,　100）
　　　　　第3頂点（−400, −200,　100）
　　　　　第4頂点（−400, −200, −200）

難易度3，胴体のくびれた部分の左側の面。側面図を中心にして作る

・四角形　第1頂点（−400,　200, −200）
　　　　　第2頂点（−400, −200, −200）
　　　　　第3頂点（−200, −100, −100）
　　　　　第4頂点（−200,　100, −100）

難易度3，胴体のくびれた部分の底の面。上面図を中心にして作る

＊

さて，最後の面の辺はすべて，いままでに作った辺と重なったはずだ。こうなると，本当にここの面を作っていないのか，それとも実際にはここに面があるのかわからなくなる（図3参照）。箱を作って，ふたの面を作るときには，必ずこういう状態になるので，比較的よくお目にかかる状態だと思う。今回は，面の数が少ないので，ここはまだ作ってない面だとか，ここは作ったとかわかると思うけど，面数が多くなるとすぐにわからなくなる。こういうときは，次のようにしたらいい。

まず，ふたになるはずの面の頂点の位置に3Dカーソルを移動させる（もちろんその頂点は，前に作った面の頂点と同じになるはずだ）。今回だったら（−400,200,−200）ぐらいでいいだろう。

そして，Bパネルの上から4行目の「点−>面」というところをクリックする。この「点−>面」には，現在の3Dカーソルの位置を頂点にしている面を順番に呼び出す，という機能があるので，ここを何回かクリックして，いまから作ろうとしている面があるかどうか確認すればいい（図3参照）。

<その5>コックピットの作成

さて，胴体の作成が終わったら，次はコックピットを作る。わずか5面しかないし，どれも難易度1，2の面ばかりなので，すぐにできるはずだ。ただ，いままで作ってきた面のアトリビュート名は「body」だったけど，これから作る面のアトリビュート名は「cockpit」なので，アトリビュートの指定の変更を忘れないように。

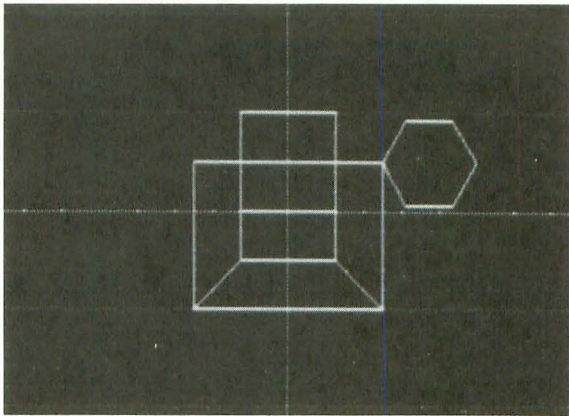



図2　正面図

＊

アトリビュート名「cockpit」

・長方形　第1頂点（　100, −100,　100）
　　　　　第2頂点（　100,　100,　100）
　　　　　第3頂点（−300,　100,　200）
　　　　　第4頂点（−300, −100,　200）

難易度2，コックピットの前の面。正面図を中心にして作るとラク

・長方形　第1頂点（−300, −100,　200）
　　　　　第2頂点（−300,　100,　200）
　　　　　第3頂点（−400,　100,　200）
　　　　　第4頂点（−400, −100,　200）

難易度1，コックピットの天井の面。上面図を使って作る

・長方形　第1頂点（−400, −100,　200）
　　　　　第2頂点（−400,　100,　200）
　　　　　第3頂点（−500,　100,　100）
　　　　　第4頂点（−500, −100,　100）

難易度2，コックピットの後ろの面。正面図を中心にして作るとラク

・四角形　第1頂点（−500, −100,　100）
　　　　　第2頂点（　100, −100,　100）

図3　面の確認方法

この面があるかどうかわからない

この点に3Dカーソルを移動して「点−>面」を実行する。面があれば点滅する。

図4　正多角形の作り方

第３項点(－300，－100，　200)
第４項点(－400，－100，　200)

難易度１，コックピットの右側の面。側面図を使って作る

・四角形　　第１項点(－500，　100，　100)
第２項点(　100，　100，　100)
第３項点(－300，　100，　200)
第４項点(－400，　100，　200)

難易度１，コックピットの左側の面。側面図を使って作る

＊

さて，これで胴体部分の作成が終わったことになる。難易度１，２の面が作れるようになったら，これだけでかなりいろんな物体が作れるはずだ。次からはいよいよコマンドを使った面の作り方について説明する。ここまでちゃんとできた人は，一度セーブしてから次に進むこと。

エンジン部分を作る

＜その１＞正多角形の作り方

いままでは，ほとんど前回の復習みたいなものだった。でも，これからは違うぞ。これから作る面は，いよいよＡパネルのコマンドを使って面を作るのだ。コマンドを使うので，すべての項点を指定する必要がないぶん操作はラクだ。でも，慣れないうちは，自分の考えていたのと違う形になったりすることもある。それに，コマンドを使って面を作ると，格子点上に項点がこないので，あとでその点に３Ｄカーソルを移動するとき面倒臭いので，多用は禁物だ。

最初は，正多角形の作り方だ。これは「紙飛行機２号」のエンジンの部分になる。だから，アトリビュートもちゃんと「engine」に変更してから，次を読むように。

まず，正多角形を作るには，正多角形の作成モードに

柚姫の明るい悩み相談室（番外編）

D-ōGA転覆す!?
バージョンアップサービスにトラブル発生！

1990年９月に行われたCGAシステムのバージョンアップサービスにおいて，重大なトラブルが発生したとの情報を得た姫は，さっそくDōGAプロジェクトルームに潜入，これより姫の恐怖の追及が始まるのであった……。

＊

ビ，ビー。静かなマンションにブザーの音が鳴り響く。プロジェクトルームのドアが細めに開いた。

姫「いったいどのようなトラブルだったのですか？」
MAX田口（以下M）「知らん，知らん」
姫「大きなバグでも発見されたのですか？」
M「知らんゆうてるやろ！」
姫「まさかウイルスでは？」
M「あほなこというな。確かにバグの問題もあるけど，要は発送上のトラブルや。それ以上は何もいえん」

MAX田口氏はそういって中に入っていった。姫はマイクを片手に，プロジェクトルームの奥深くへとさらに進んでゆく（BGM：インディジョーンズのテーマ）。

姫は事件当時の発送担当者はFFE三保氏であることをつきとめ，インタビューに成功した。

FFE三保「僕の責任じゃないよー。コピーの担当者が悪いんだ。コピーの担当は島さんだよー」

どうやら，発送のトラブルとはコピーミスらしい。次に姫はroot島氏をつかまえた。

姫「“コピーの担当者が悪いんだ”という声が聞かれているのですが」
root島（以下r）「つまり，今回のバージョンアップサービスで２枚のディスクが送られたんやけど，その２枚がまったく同じやったという苦情がたくさん来たんや」
姫「なぜそんなことが起きたんでしょうか」
r「さあ，なんでかさっぱりわからんなー。まったく悪魔の仕業としか……」
（むこうでかまた氏とおぼしき人のくしゃみが聞こえる）
姫「どの程度の苦情が来たのですか？」
r「１～２割だったから，相当な数やな。でも，コピーミスの苦情だけやないで。なんでもMAILに“このバージョンはディスク３枚組だ”とか“IOCS.Xの新しいバージョンが入っている”とかウソばっかり書いてあって，だいぶ混乱を招いたんや」

どうやら，トラブルの根は深いようだ。さらにマスターディスク作成担当レイバー小林氏にインタビューをした。

姫「マスターディスクに問題があったそうですが……」
レイバー小林（以下レ）「エ～，そーなんかなぁ」
姫「はぁ」
レ「システムが２枚とサンプルが１枚で，バージョンアップしたのはシステムなんやから数はあってるやん。IOCSははじめ入れるつもりやったんやけど，シャープにいわんと，勝手にでけへんからなぁ」
（電話すればいいことなのにと思う姫）
姫「今後はIOCSを入れていくんですか」
レ「入れるつもりやけど，シャープにまだ確認取ってないから，なんともいわれへんなぁ」
（やっぱりさっさと電話すればいいのにと思う姫）
レ「どっちにしてもあのバグから比べたら，たいした問題とちゃうやん」

どうやら，大きなバグもあったようだ。バグ出し隊員のGAVAN島田氏を追求してみよう。

姫「バージョンアップ版に大きなバグがあるそうですね」
GAVAN島田（以下G）「FFE（モーションデザインツール）がぜんぜん動かないみたいやね。HANIM（ハイスピードアニメーション）は動くけど，そのために必要なCRD（色数制限ツール）がバグってて，画面の下のほうに黒い帯が発生するそうだ」
姫「それって，ほとんどバージョンダウンっていいません？　何かコメントお願いします」
G「バグ出しはやったんだけど，バグ出しするあとからあとからバージョンアップしたので，追いつかなくてー（更新矢の如し）。だいたい土壇場になって，コンパイラ変えたりするか？」

どうも今回の事件は責任がはっきりしない。責任者であるかまた氏に話を聞いてみることにした。

姫「いったいなんでそんなことになったんですか」
かまた（以下か）「わたしゃ，ずっと旅行しとったから，今回はノータッチなんやけどなぁ。いや～，スイスの山はやっぱりきれいやな。朝焼けのマッターホルンは感動ものやで。夜明け前から山登って……」
姫「そんな話は聞いていません。それで，今後の対処というのは？」
か「やっぱり，バージョンアップサービスのバージョンアップサービスを近日中に行わないかんやろなぁ。ディスク１枚を無料で送るつもりやわ」
姫「わかりました。責任者として，今回のトラブルについてどう思われますか？」
か「このような遺憾な事態となったのは，まったくイカンなぁ」
姫「……ありがとうございました」

＊

このたびはたいへん混乱を引き起こし，申し訳ありませんでした。上記の内容は冗談（妙にリアリティのある冗談？）として，関係者一同，深く，深～く反省しております。バージョンアップサービスのバージョンアップサービスがお手元に届くまでいましばらくお待ちくださいね。また，９月から11月にかけて，今回のトラブル以外にも，試験や学祭があったため，トラブルに関する問い合わせや苦情にぜんぜん対応できなかったことをお詫びします。

それにしても，当プロジェクトルームには，毎日多数の問い合わせが来ます。「CGAシステムはまだ手に入りますか？」などというのはかわいいもので，なかには，とっくに締め切られている前回のCGAコンテストのビデオのお金を勝手に振り込んでおいて，「まだ来ない！」と苦情をいうような困った方もいます。基本的に，当方に問い合わせて返事がなかったら，DōGA内でトラブルが発生して，対応できない状態に陥っているのだと好意的に受けとめてください。できるだけのことはやっているのですが，まぁ今年，留年した５人のスタッフに免じて，お許しを……。

しないといけない。Aパネルの下から4行目の「正多角形作成」のところをクリックすればAパネルが正多角形の作成モードになる。今回作るのは，正六角形なので，まずAパネルのいちばん上の行の「角数　12」と表示してあるところで，マウスの左ボタンを6回クリックして角数を「6」にする。次に3Dカーソルを（−500,300,100)の点に移動させる。ここで，スペースキーを1回押す。もちろんスペースキーの代わりに，Bパネルの「点確定」をクリックしてもいい。この点が正多角形の中心，正確にいうと正多角形に外接する円の中心になる。そして，3Dカーソルを正面図で左に2分の1マス分(−100)だけ移動させて，もう一度スペースキーを押す。この点は，正多角形の大きさ，これまた正確にいうと，正多角形に外接する円の半径になる。また，正多角形の頂点が，この点にくるように作られる（図4参照)。そしてさらに，3Dカーソルを正面図で上に1マス分（+200）だけ移動させて，スペースキーを押す。この点は，実をいうと（今回は)，正面図上の点だったらどこでもいいのだ。この点の役目はただひとつ。中学校のときに平面を決定するには最低3点が必要だと習ったと思う。つまり，最後の1点はできる正多角形がどの面の上にあるかを決定するためのものだったのだ。この点を適当に変えると，傾いた正多角形も簡単にできるようになっている。3つの点が確定したら，リターンキーを押す。すると，図4のような正六角形が表示されるはずだ。そして，正多角形ができたら，ESCキーを押して，正多角形作成モードを終了する。

<その2>角柱の作り方

次は，さっき作った正六角形を底面とする角柱を作ることにしよう。まず，Aパネルの上から5行目の「角柱作成」のところをクリックして，角柱作成モードに入る。そして，3Dカーソルを（−500,200,100）に移動させて，スペースキーを押す。そして，次に3Dカーソルを側面図で左に3と2分の1マス分（−700）移動させて，もう一度スペースキーを押す。このとき，座標は（−1200,200,100）になっているはずだけど，もしなっていなかったら，確定を取り消してから正しい点で点確定すること。そして，ちゃんと2つの点を確定したら，リターンキーを押す。すると，さっき作った正六角形を底面として，高さがいま指定した2点の距離となるような角柱ができるはずだ。失敗した人は……いちいち面を全部消すか，セーブしたところからやり直すしかない。

角柱作成で指定した2点は，角柱の高さと傾きだということはすぐにわかったと思う。もちろんこれを使うとピサの斜塔なんかも簡単に作ることができる。指定する直線をちょっと傾かせればいいだけだ（図5参照)。

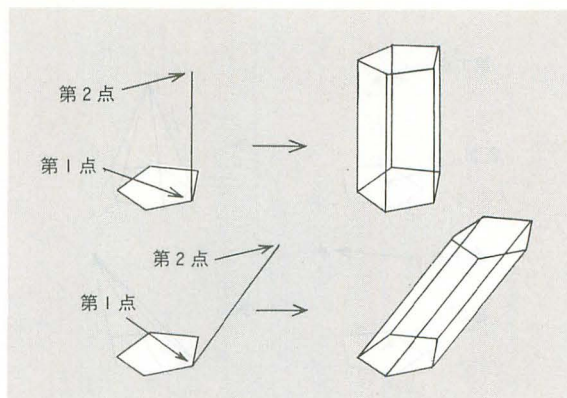



図5　角柱の作り方

ただ，この角柱作成で注意する点は，底面の指定だ。角柱作成モードでは，底面を作るとか，底面を指定するということができない。そこで，角柱作成では，底面は現在の面ポインタがある面になるのだ。つまり角柱を作るには，あらかじめ底面を作っておかないといけないということになる。そうするとあとは，高さのベクトルを指定するだけで，側面と，上面を作ってくれるのだ。

さて，さっきの解説で聞き慣れない言葉が出てきたと思う。面ポインタなんか，「そういやマニュアルに載ってたな」，ぐらいしか覚えてない人がほとんどだろう。ポイント面と，各コマンドの関係は，CADを使ううえでとっても重要なことだけど，今回は完璧に理解なんかしなくていい。詳しくはマニュアルを読めばいいとして，とりあえずは次のように覚えておけばいいのだ。「(面を作ると，その面に面ポインタが移動するので）普通，面ポインタは最後に作った面にある」。だからさっきは，正六角形を底面とした角柱ができたわけだ（最後に作った面がエンジン部分の正六角形だったからね)。そこで，角柱を作るときは，底面を作って，すぐに角柱を作るようにして，底面の作成と角柱の作成を，ワンセットでやるようにする。そうすれば，面ポインタなんかまったく気にしなくていいぞ。

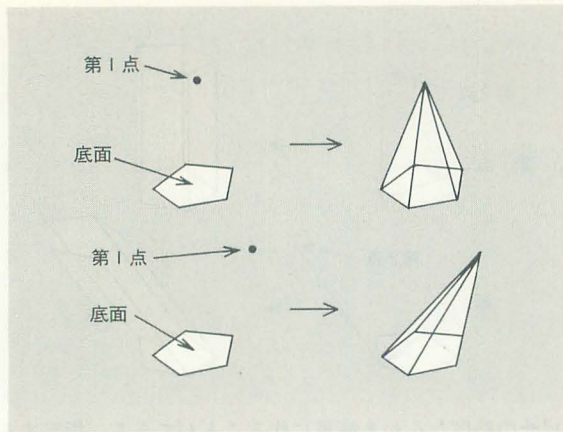
あとは，同じ角柱を正面図で下に1と2分の1マス分(−300)だけ移動したところに作ればエンジンの部分は出来上がりだ。まず，中心が（−500,200,−200)で，横に2分の1マス分（100）動かした点を頂点に持つ，正面図に平行な正六角形を作る（長い文章だなぁ)。そのあと，すぐに高さが3と分の1マス分（700）の角柱を作るのだ。これが終わったら，もう一度セーブしてから次に進もう。

各読者通達事項

●CGAコンテスト事務局より

10月15日の新聞各紙でも報じられ（小さかったから誰も気づかなかっただろうなぁ)，ますます盛り上がるCGAコンテストの審査員が新たに2名加わりました。映像作家の岩井俊雄氏とアニメーション作家の古川タク氏です。これで，審査員は総勢11名という豪華さです。

さて，締め切り（12月31日）まで残すところあと2週間足らずとなったわけですが，皆さんラストスパート頑張ってますか？　“完成しそうにないから，また来年のコンテストを目指そう”などと思ったら大きな間違いですよ。そんなことを考えていると，作品はいつまでたっても完成しません。来年になれば，コンテストのレベルも上がるし，同じネタ（作品）に1年以上つき合ってくると，嫌になって，完成せずに次の作品を手がけてしまうものです。与えられた期限を守るというのは，作品制作の重要なポイントです。必要ならば，ストーリーを短くしたり，もっと簡単にできる表現に切り替えるなどの対応も必要となるでしょう。あとはもう努力と根性で，除夜の鐘がなるまで，連日の徹夜，頑張ってください。

図6　角錐の作り方

翼の部分を作る

<その1>翼を作る

翼の部分は，難易度2の面を2つ作るだけなのですぐにできると思う。ただ，何度もいうようだけどアトリビュートの変更だけはお忘れなく。

＊

アトリビュート名「wing」

・四角形　第1頂点(−1000,　1000,　400)
　　　　　第2頂点(　−700,　1000,　400)
　　　　　第3頂点(　−500,　 400,　100)
　　　　　第4頂点(−1000,　 400,　100)

難易度2，左上の翼。側面図を中心にして作るとラク。また，Y座標の移動は，この面は上面図に入りきらないので正面図を使ってするといい。

・四角形　第1頂点(−1000,　1000,−500)
　　　　　第2頂点(　−700,　1000,−200)
　　　　　第3頂点(　−500,　 400,−200)
　　　　　第4頂点(−1000,　 400,−500)

難易度2，左下の翼。側面図を中心にして作るとラク。また，Y座標の移動は，この面は上面図に入りきらないので正面図を使ってするといい。

<その2>角錐の作り方

最後にミサイルの部分を作る。まずアトリビュートを「missile」に変更したら，正面図に平行な面を作る。

＊

・正方形　第1頂点(−1100,　1000,　400)
　　　　　第2頂点(−1100,　1100,　500)
　　　　　第3頂点(−1100,　1100,　500)
　　　　　第4頂点(−1000,　1000,　400)

難易度1，超簡単な面。これがこれから作る角錐の底面になる。

＊

角錐の作成では，角柱のときと同じで面ポインタのある面（つまり最後に作った面）が底面になる。ただ，角柱のときと違うのは，指定する点が1点，つまり角錐のいちばん上の点（なんていうのか忘れた）を指定するだけで，角錐ができるということだ（図6参照）。この場合，点を確定するつもりで，うっかりスペースキーを押してしまうと，有無をいわさずそこを頂点とする角錐ができてしまう。間違えた点でスペースキーを押してしまったら，いちいち面を消さないといけない。だから，角錐の作成は十分注意して行ったほうがいい。

今回は，先ほど作った面を底面とした角錐を作る。ま

モデラー高津のLOGIN

「好きやねんCG大阪合宿」のレポートの中にも少し話が出ていましたが，DōGAでは現在，オリジナルのウィンドウシステム「Ko-Window」を開発しています。なぜ，CGAの団体であるはずのDōGAがウィンドウシステムを開発しているかというと，単に趣味の世界です。しかし，来年完成予定のDōGA CGAシステムVer.3はこのKo-Window上で動作させるかもしれません。J&PのSIGでは，常にKo-windowに関する最新の情報をアップしていますので，ぜひご覧ください。

Ko-Windowの利点と欠点は以下のとおりです。

利点
・仕様が公開されている。
・プログラムをCで記述できるので，開発が容易である。
欠点
・動作が不安定。
・ころころとバージョンが上がっている。
・イベント駆動型（注1）なので，プログラムが組みにくい。

現在，Ko-Windowでは以下のようなアプリケーションが動いています。

HELLO.WIN	ご存じハローワールド
DCLOCK.WIN	デジタル時計
PED.WIN	パターンエディタ
FINDER.WIN	ファイルの選択および移動を行うプログラム
ACLOCK.WIN	アナログ時計
CALC.WIN	電卓
X.WIN	Xのロゴを表示する（注2）
EYES.WIN	マウスカーソルを見つめる目玉（注3）
BACK.WIN	背景を変える
COMMAND.WIN	ウィンドウ版Command.X
POPUP.WIN	ポップアップメニューを実現する
WED.WIN	テキストエディタ
FREE.WIN	メモリの空き容量の表示するプログラム
SLIDE.WIN	CGAシステムのPICファイルを表示する
GRAPH.WIN	グラフィック表示のデモ
REND.WIN	CGAシステムのレンダラー
MAP.WIN	メモリの使用状況を表示するプログラム

なお，このKo-Windowは，J&P以外のネット（注4）でも少しずつ，公開していきます。

注1
普通のCのプログラムでは，main関数側から処理に必要な関数を呼び出していきます。Ko-Window用のプログラムでは，関数の呼び出しをウィンドウシステムに任してしまい，用事ができたら関数を呼んでねっ，といってmain関数を終了します。すると，ウィンドウを描き直すとか，キーボードから入力があったとか，なにか処理が必要になったらその関数を呼んでくれるわけです。普通のプログラムはこちらからてきぱきと能動的に処理をするわけですが，イベント駆動型の場合，むこうからしてくれと頼まれたことだけをするという受動的な処理をするわけです。

注2
X-WINDOWで有名なものに似ているけど字体が違います。

注3
X-WINDOWのデモでよくあるあれです。

注4
現在DōGAでサポートできるネットは，
J&P HOTLINE
ASCII NET PCS
PC-VAN
NIFTY Serve
SPSネット
の5つです。

＊　　＊　　＊

今月のアップデータ

"Gate open. Gate open. fire!"

なつかしのウルト○ホーク1号です。ちゃんと3機に分離します。制作は，CADを使わせたら日本一（？），チームTOSAKAです。ただ，データが大きすぎるので，3機いっぺんにCADで読み込もうとすると，バスエラーが出るというとんでもない作品です。

ずは，Aパネルの「角柱作成」の下にある「角錘作成」のところでマウスをクリックして，角錘作成モードに入る。そして，3Dカーソルを（100, 1000, 400）のところに移動させて，リターンキーでも，スペースキーでもどちらでも好きなキーをどちらか1回だけ押す。これだけだ。これだけで，あっという間に角錘が出来上がるのだ。もっとも，これはちゃんとポインタ面がさっき作った面にあったらの話で，もし変なことをして，ポインタ面が移動していたら，予想もつかない面（というかポインタ面がある面）を底面とした角錐ができる。だから，角錐を作るときも，角柱を作るときと同様に，底面の作成と角錐の作成はワンセットでやったほうがいい。

あとはもうひとつ作るだけだ。まずは，底面を作る。

＊

・正方形　第1項点(−1100,　1000, −500)
　　　　　第2項点(−1100,　1100, −500)
　　　　　第3項点(−1100,　1100, −600)
　　　　　第4項点(−1000,　1000, −600)

難易度1，超簡単な面。これが左下の角錘の底面になる。

＊

そして，面を作ったら，すぐに角錐作成モードに入って頂点を（100, 1000, −500）とする角錐を作る。これが終わったらセーブして，そして，CADを終了する。

ついに完成だ

いままでに作ったのは，「紙飛行機2号」の左半分だけだった。でも，この「紙飛行機2号」は，左右対称なので，あとはMIRRを使って左右反転コピーすれば右半分の部分が自動的に作られて，「紙飛行機2号」が完成する。MIRRでは，オプションをつけずに反転コピーすればいい。ただ，万が一間違えたときのために，出力ファイル名は，元のものと違う名前にしておいたほうが無難だ。MIRRを使ったら，もう一度CADを立ち上げて，見てみよう。ちゃんとできていたら成功だ。今回はCADの特集なので，これで終わりだ。できたら，これを使って，かっこいいカットを作ってほしい。もし，かっこいいカットができなくてもモデル（紙飛行機2号）のせいにはしないように。

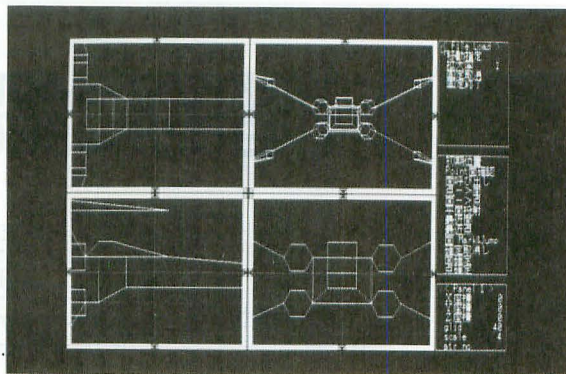



図7　完成時の形状ファイル

おわりに

CGAシステムが使えないという人の約8割は，CADでつまずいているといわれている(当社推定)。CADに慣れるには，1時間でも多くCADを使うことだ。今回作った「紙飛行機2号」なんか，慣れると5分ほどでできてしまう。実際私はこれを5分ほどで作ったのだから間違いない。でも，その5分でできる物体を作るのを説明するのに，いったいどれだけかかったか……。

ここ数カ月，隔月連載といったわりには毎月連載をしているような気がするが，次回は，かまたさんの“実用的な構造体の使い方「戦えロボット君2」”をお送りできるはず，だろう。

寺田の教育的指導

今月は私が「律速段階」です（DōGA用語でいちばん原稿が遅れている者をこう呼ぶ）。だから，あわてて文章を書いてます。文脈も変になりそうです。試験の真っ最中ということで許してやってください。

以前このコーナーでご紹介した森山知巳さん（あの本職の画家の方です。忘れた人はバックナンバーをひっくり返してみてください）から再びディスクが届きました。今度はなんとディスク5枚組です！　この量だけでも期待が高まりますね。さぁて，さっそく作品を見てみましょう。

むむ，こ，これは……。あの「人体モデル」が帰ってきた!!（んな大袈裟な）あのままで終わるはずがないと思っていましたが，やはり作品としてまとめる方向で制作されていたようです。

同封の手紙によると（いつも丁寧なお手紙ありがとうございます），タイトルは「SWORD」，テーマは「時の流れと人」だそうです。うーむ，なかなか凄そうですね。ストーリーは，サンプル版ということでよくわからないのですが，「人」が家来（？）の「犬」（フレンダーそっくり）とともに「ドラゴン」と戦うというものらしいです。

以前も書きましたように，作品に技術的な問題点はとくにありませんし，手紙に『動きなどのノウハウをお教えください』とありますので，そのあたりについて書いてみたいと思います。とはいっても私も映像のプロというわけでもありませんので，DōGAのほかのスタッフの意見も参考にして，気づいた点を挙げてみます。

まず，犬がドラゴンに飛びかかるシーンですが，少しカット割りが細かすぎる気がします。なかなか凝ったシーンで，犬が地面を蹴ってドラゴンに飛びかかるまでの間に，違う方向から見たカットが入るのですが，その時間が短いので「目がちらちらする」ような感じを受けます。もう少し1つひとつのカットを長くしたほうがよいのではないでしょうか。

それと，飛びかかるときに犬の顔がドラゴンとは別のほうを向いていますが，これは不自然だと思います。犬の闘志を表現するためにも，顔はまっすぐドラゴンを睨みつけていたほうがよいでしょう。フレームファイルで『vec( )』を使えばうまくいくと思います（顔の向きを，顔からドラゴンへ向かうベクトルで与える）。

また，ドラゴンに跳ねとばされた犬に人が走り寄るシーンで，倒れた犬を人が見つめているカットがありますが，このカットから次のカットへの切り替えが少し早いと思います。その次のカットで人が怒りのポーズをとっていることから考えて，このカットでは犬が殺された（あるいは怪我をした？）ことに対する「人」の悲しみ・悔しさが表現されているのだと考えられます。だとすれば，一般的にいって，もう少し長いカットでないと観ていて感情移入がしにくいのではないでしょうか。

以上，かなり偉そうな書き方になってしまいましたが，一応常識的な線で判断して書いてみました。参考になれば幸いです。まだサンプル版ということなので，完成版に期待しています。頑張ってください。

ところで，「MASAYA」シリーズはいまのところボツということですが，ちょっと残念です（個人的にはこちらのほうが好きだったのです）。あのメリーゴーラウンドも気に入ってたんですが……。暇があったらまた続きを作ってください。楽しみにしています。

それではまた次回に。じゃあ！

清水和人流
プログラミング道場
〔その3〕

段階的に鍛えていくべし

Shimizu Kazuto

みなさんお待たせいたしました。清水和人流プログラミング道場の3回目の講義です。今回はスプライトを使ったテーブルテニスゲームを作りながら，プログラミング後生説というものを理解していただくのが狙いです。

さて，今日も1日お疲れさま。デモ，寝る前にちょっと愛機を立ち上げる，その姿勢が未来の大物を育てるのである。なにせBASICで遊べる程度のプログラム力があれば，将来さまざまなソフト関係の仕事をこなせるからである。偉大な教育の場，それこそがBASICの世界なのである。

逆に見るとこんなふうにも考えられる。BASICで遊べる程度の腕があれば，多少人格的にはフラフラしていてもなんとか食える世の中なのである。新人類という言葉が聞かれるようになってから，社会人といえども立派な人は日本には少なくなってしまったのである。だからといって学校の勉強をなまけてみたりすると，これがまた食えなくなるから，ヤル気はなくさないように。

まあ，とにかく寝る前に愛機を立ち上げたということで，私は今日も今日とてその前に座ったのである。何をしようという気持ちもまだない。いわば無の状態である。それではこの立ち上げが無意味かというと，そうでもない。それは心の中に「何かやろうかな」という小さな火を灯す。この小さなきっかけが大切なのである。これで次第に好きになって，病みつきへの道は遠くない。だいたい，BASICで挫折する人は遠大な計画を立てすぎるのである。凄いプログラムを組もうと思って，あれもこれも思いつく機能はすべて取り込む，なんてやっているとバグのオバケになってしまい，「もう二度とBASICなんて」と別離を決心するハメになるのだ。そりゃ，X68000は立ち上げに時間がかかるし，立ち上げたらTVも見られない。しかし，夜にプロ野球ニュースを見るより，もう少しはマシなのではないだろうか。

1.今日はスプライトから

X68000を買った人には，カタログや広告のさまざまな宣伝文句につられて買った人も多いだろう。でも，その機能を自分で使った人は案外少なそうである。フルカラーグラフィック，FM音源，AD PCM，スプライト機能……etc，そういえばどれも昔のパソコンにはない，新しい用語（もうかなり古いが）ばかりである。

これらの機能は限定つきでBASICから使用が可能である。さて今日は何をやろうかな。んー，スプライト，これなら簡単そうである。スプライトといっても飲み物ではない（あー，言ってしまった）。RPG（ロールプレイングゲームだよ）に出てくる例の妖精ってやつである。お花畑をちらちら飛び回る。これがスプライトである。画面内を自由に動かせるキャラクタ，これがX68000スプライトである。

スプライトの使い方を1つひとつ説明するのは，ここでの目的ではない。それはBASICマニュアルに任せるとして，まずはサンプルプログラムから入ろう。リスト1はBASICマニュアルのパクリである。実に味わいのないサンプルであるが，一応スプライトのなんたるかをすべて含んでいる。スプライトをどう使うかは，このリストとマニュアルの各命令を見比べて，何をやっているか見ればいい。このくらいの努力を惜しんでいては将来は閉ざされたといってよかろう。

これを走らせてみると，「こらまったどうじゃろかい」，遅いのである。「なんやスプライトちょうても，こんな遅いのかいな」と関西オヤジの登場である。まあBASICだからということもあるが，そのほかにも2つの遅い理由がある。

1) 座標を1ドットずつ動かしている
2) SIN関数の計算が遅い

1)については，動かすのに2ドットごとにすれば2倍の速さになるし，2)についてはなにもSIN関数なんか使ってまでスプライトさせなくたってよいのである。とにかく，ドキッとさせられるサンプルだ。

そこで，もっと単純なスプライトを見るために，とりあえずリスト2を作ってみた。

●リスト1

```
 10 int ai,x,y
 20 dim char SP1(255)={
 30   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
 40   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2,
 50   0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 3,
 60   0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 4, 4,
 70   0, 0, 0, 0, 0, 0, 1, 2, 2, 3, 3, 4, 4, 4, 5, 5,
 80   0, 0, 0, 0, 0, 1, 2, 3, 3, 4, 4, 5, 5, 5, 6, 6,
 90   0, 0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 6, 6, 6, 7, 7,
100   0, 0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 7, 7, 7, 8, 8,
110   0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 7, 8, 8, 8, 9, 9,
120   0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 9, 9,10,10,
130   0, 0, 1, 2, 3, 4, 5, 6, 7, 9, 9,10,10,10,11,11,
140   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,10,11,11,12,12,
150   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,12,13,13,
160   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,14,
170   1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,15,
180   1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,15}
190   screen 1,3,1,1
200 sp_clr(0,255)
210 sp_off(0,127)
220 sp_disp(1)
230 sp_def(0,SP1)
240 for ai=1 to 15
250    sp_color(ai,rgb(ai*2,ai*2,ai*2))
260 next
270 for ai=0 to 384
280    x=ai*2-16
290    y=(sin(2*3.1416##*x/256))*200
300    sp_move(0,x,255-y,0)
310 next
320 goto 270
```

これはアクビが出るほど簡単だが，画面の中を四角が飛び回るのである。これならどうやらスピードも許せるようだ。ちなみに，220，230行のdx，dyの値を大きくすればスピードは速くなるし，小さくすれば遅くなる。

このプログラムはBREAKしないかぎり，ずっと動いているので，動かしたままにして冷たいものでも飲もう。最近のお勧めは見えないゼリードリンクのアップルである。ちらっと沈黙の艦隊に目を通すのもよかろう。で，目を上げるとまだスプライトしている。ずっと見ているうちに「これゲームになんないかな？」と考える。

あっ！

そう，まさにこれと同じような経過をたどってゲームのヒントを思いついた人が，なんと20年前，江戸時代の末期にいたのである。そしてそれがTVゲームの元祖となった「テーブルテニス」である。

いまでこそファミコン，ゲームボーイを筆頭とするコンピュータゲームが全盛であるが，その先駆けとなったゲームマシンがあったのである。それは家庭のテレビにつなぐとテーブルテニスが楽しめるという，いまから考えるととんでもなく低レベルなものだが，当時若干売れた（現に私の友達も買った）。

リスト2はまさにこのゲームの基本なのである。これの左右にパドルが現れるともうそれだけでゲームなのだ。

ここまでくるとパドルを動かさなくてはならない。しかし，2人用ときて，また，

あっ！

●リスト2

```
 10 int ai,x,y,dx,dy
 20 dim char SP1(255)
 30 /* initialize
 40 screen 1,3,1,1
 50 /* sprite pattern definition
 60 for i=0 to 255:SP1(i)=1:next
 70 sp_clr(0,255)
 80 sp_off(0,127)
 90 sp_disp(1)
100 sp_def(0,SP1)
110 sp_def(1,SP1)
120 sp_color(1,rgb(30,30,30))
130 /* draw wall
140 ic1=rgb(30,30,30)
150 box(0,0,511,15,ic1,&HFFFF)
160 paint(256,8,ic1)
170 box(0,496,511,511,ic1,&HFFFF)
180 paint(256,504,ic1)
190 /* initial position & velocity
200 x=255
210 y=255
220 dx=6
230 dy=4
240 if y<=16 then dy=-dy
250 if y>=480 then dy=-dy
260 if x<=16 then dx=-dx
270 if x>=480 then dx=-dx
280 x=x+dx
290 y=y+dy
300 sp_move(0,x,y,0)
310 goto 240
```

である。X68000のBASICでは同時に2つのキーセットができないのである。つまり2人の入力を同時に判断できないのだ。この逃げ道として最も安易な方法が「ジョイスティックを使う」である。X68000のユーザーなら，ジョイスティックを1つくらいもっていそうである。2つとなると苦しそうなので片一方はキーボードだ。こうして，出来上がったのがリスト3である。

リスト3はまさに「テーブルテニス」の

●リスト3

```
 100 int ai,x,y,dx,dy,in1,in2,ip1,ip2,ipmax=5
 200 str in1$,in2$
 300 dim char SP1(255)
 400 /* initialize
 500 screen 1,3,1,1
 600 /* sprite pattern definition
 700 for i=0 to 255:SP1(i)=1:next
 800 sp_clr(0,255)
 900 sp_off(0,127)
1000 sp_disp(1)
1100 for i=0 to 10:sp_def(i,SP1):next
1200 sp_color(1,rgb(30,30,30))
1300 /* draw wall
1400 ic1=rgb(30,30,30)
1500 box(0,0,511,15,ic1,&HFFFF)
1600 paint(256,8,ic1)
1700 box(0,496,511,511,ic1,&HFFFF)
1800 paint(256,504,ic1)
1900 repeat
2000 ip1=0:ip2=0
2100 repeat
2200 locate 28,1:print ip1;:locate 31,1:print "-";:print ip2
2300 /* initial position & velocity
2400 if x<0 then dx=24 else dx=-24
2500 x=255 :y=215         :dy=16
2600 x1=0  :y1=215:dx1=16:dy1=24
2700 x2=496:y2=215:dx2=16:dy2=24
2800 /* start
2900 repeat
3000 /* boundary condition
3100 if y<=16 then dy=-dy
3200 if y>=480 then dy=-dy
3300 /*if x<=16 then dx=-dx:goto 4300
3400 if x<=16 then {
3500     if y> y1-24 and y<y1      then dx=16 :dy=-32
3600     if y>=y1    and y<y1+16   then dx=abs(dx):dy=-24
3700     if y>=y1+16 and y<y1+32   then dx=abs(dx):dy=-16
3800     if y>=y1+32 and y<y1+48   then dx=abs(dx):dy=0
3900     if y>=y1+48 and y<y1+64   then dx=abs(dx):dy=16
4000     if y>=y1+64 and y<y1+80   then dx=abs(dx):dy=24
4100     if y>=y1+80 and y<y1+104  then dx=16 :dy=32
4200 }
4300 /*if x>=480 then dx=-dx:goto 5300
4400 if x>=480 then {
4500     if y> y2-24 and y<y2      then dx=-16:dy=-32
4600     if y>=y2    and y<y2+16   then dx=-abs(dx):dy=-24
4700     if y>=y2+16 and y<y2+32   then dx=-abs(dx):dy=-16
4800     if y>=y2+32 and y<y2+48   then dx=-abs(dx):dy=0
4900     if y>=y2+48 and y<y2+64   then dx=-abs(dx):dy=16
5000     if y>=y2+64 and y<y2+80   then dx=-abs(dx):dy=24
5100     if y>=y2+80 and y<y2+104  then dx=-16:dy=32
5200 }
5300 /*key or joystick sense
5400 in1$=inkey$(0)
5500 /*in1 =stick(2)
5600 in2 =stick(1)
5700 if in1$="8" and y1>=16  then y1=y1-dy1
5800 if in2 = 8  and y2>=16  then y2=y2-dy2
5900 if in1$="2" and y1<=416 then y1=y1+dy1
6000 if in2 = 2  and y2<=416 then y2=y2+dy2
6100 /*movement of ball and pads
6200 x=x+dx
6300 y=y+dy
6400 for i=1 to 5:sp_move(i,x1,y1-16+16*i,i):next
6500 for i=1 to 5:sp_move(i+5,x2,y2-16+16*i,i+5):next
6600 if x>=0 and y>=0 then sp_move(0,x,y,0)
6700 until x<0 or x>496
6800 /*point incriment and display
6900 if x<0   then sp_move(0,0,y,0)   :ip2=ip2+1
7000 if x>496 then sp_move(0,496,y,0):ip1=ip1+1
7100 locate 28,1:print ip1;:locate 31,1:print "-";:print ip2
7200 repeat
7300 locate 25,30:print "(c)ontinue";:in1$=inkey$
7400 until in1$="c"
7500 locate 25,30:print "          "
7600 until ip1>=ipmax or ip2>=ipmax
7700 /*display winner
7800 locate 20,1
7900 if ip1>ip2 then print "won by player 1(";ip1;"-";ip2;")"
8000 if ip1<ip2 then print "won by player 2(";ip1;"-";ip2;")"
8100 repeat
8200 locate 22,30:print "(c)ontinue or e(x)it";:in1$=inkey$
8300 until in1$="c" or in1$="x"
8400 locate 22,30:print "                    "
8500 until in1$="x"
8600 screen 1,3,1,1
8700 end
```

最も簡単なバージョンでもある。逆にいうとあまりにも味気ないバージョンでもある。でもまあこんなリストでもいろいろ苦労はある。

1) IF文が増えたためにスピードが落ちた。このため，何ドットも飛ばして動かすことでスピードアップさせざるを得ない(BAStoCコンパイルの助けは借りないのだ)

2) パドルと玉の速度差が微妙。とんでもなく難しくなったり，簡単になったりする

3) 変化を持たせるために，パドルの当たる位置によって跳ね返る方向を変えた

こんなものでも結構1時間以上かかるものである。もう寝よう。1日のうちにあまり深入りすると，アキがくる。自分をアキさせないことが大切なのである。寝ることによってまたよいアイデアが湧くかもしれない。

2.改造はアイデア次第

さて，自分で作ったゲームのよさはいくらでも改造できることにある。市販のソフトはいくらよくできていても，この改造に関してはほとんど無力である。ゲームは楽しめても自分の想像力を生かすところがない。自作なら，たいしたことのないゲームでも遊べると同時に自分の想像力を生かし，またプログラム力をアップすることができる。この差は歴然としている。

そして自作用プログラムに最も適した言語がBASICなのである。自作用言語こそがBASICであり，しかもX68000にはBASICからC言語への変換まで用意されている。そのため，多少普通のBASICと異なるが，なあに，Cのインタプリタだと思えば，こりゃええと思えてくる。そこまでめぐまれたX68000，しかもおおかたの機能はBASICでサポートされているとくれば，もうゲーム機だけにしておくのはナンセンスである。

さて，昨日作ったこの単純版「テーブルテニス」，どんなふうに改造してやろうかと1日中考える。「これは速いけど難しそう」，「これは簡単に変えられるけどいまいち」といったアイデアが次々と湧いてくる。これがプログラム後生説である。

プログラムの機能や使用をあらかじめ詳細に決め，準備を整えてからコーディングにかかる。このやり方を前生説とすれば，簡単な雛形をまず作っておき，それに次々と機能を加えていって大きくふくらませていくやり方が後生説である。このやり方はソフトハウスの人にはお勧めできないが，一般パソコンプログラマには最適な方法である。少しずつ作るのでその都度試すことができ，デバッグは簡単になるし，思いついたことをどんどん形にしていける。今回のテーマはスプライトだが，第2のテーマはバージョンアップ，すなわち改造をしていくことである。

1) 改造その1 パドルの前後の動き

まず最初に思いついたのが，「パドルを前後に動かせないか」ということである。テニスでいえば「前に出る」，これができるとゲーム内容もぐっと深まるのではないか。

この改造はパドルを動かすこと自体は簡単である。Y座標だけでなくX座標も変えてやればよいのである。それをテンキーやジョイスティックの前後方向に対応させればよい。ついでに斜めにも動かせるようにしたい。

気をつけるのは前後の境界処理である。ある場所以上には進めないように，X座標で判断する必要がある。それと，玉がパドルに当たったかどうかの判断が大きく変わる。いままでは後ろの境界でのみ判断すればよかったが，この改造でパドルのX座標に合わせて判断することが必要になった。

そんなことを考えつつ，改造した修正部分がリスト4である。これをリスト3の上に打ち込めば改造完了である。元の行を変えたところや，間に挿入したところもある。そのためにリスト3では行番号を100おきにリナンバし，改造しやすくしておいたのである。

この改造部分において2600から2700行のdx1, dx2を変えることによって，前後の動きの飛び幅を変えられる。極端にdx1=dx2=200とすれば前後2カ所をワープするだけになり，スムーズな移動とはまた違った楽しみ方ができる。

2) 改造その2 カベ現る

このタイプのゲームにつきものなのが，玉を跳ね返す「カベ」である。これをいろんなバリエーションで置くことによって玉の行方がわからなくなり，ゲームはさらに複雑になる。といっても，「カベ」だらけにしてしまうと難しすぎて面白くなくなってしまう。そのへんの呼吸が大切である。

カベをある一定の形に置いてしまうと，何回かやると飽きてしまう。そこで，1ゲームごとに乱数を用いてカベを出現させ，バリエーションを楽しむことにしよう。

リスト5がこのための改造部分である。リスト3にこのリストを重ねるとカベのあるゲームが楽しめる。

ここでのポイントはやはり玉がカベにぶつかる処理である。その位置にカベがあるかどうかは座標を比べるしかないので，少々面倒である。もう少し便利なコマンドがほしいところだ。

もうひとつのポイントはどのくらいカベを作るかである。ここでは画面の中央部，2つのパドルの中間のみに出現させた。それもあまりどぎついものでなく，ちょっとだけ出したほうが，ゲームの本質を保てていいようだ。

プレイ中に現れたり，動くカベというのも考えられるが，ここは深入りせずに別の機能を考えよう。

3) 改造その3 玉を2つに

さて，このゲームを一気に難しくしてしまおう。玉を増やすのである。これの基本的なプログラムはいままでのアルゴリズムと一緒である。ただループ1回につき2つの玉を別々に動かせばいいのである。意外に簡単な改造ですむ(リスト6)。

●リスト4

```
3410 }
3450 if x>=x1-16 and x<=x1+16 then {
4410 }
4450 if x>=x2 and x<=x2+32 then {
5720 if (in1$="7" or in1$="9") and y1>=16 then y1=y1-dy1
5740 if (in1$="1" or in1$="4" or in1$="7") and x1>=15
     then x1=x1-dx1
5800 if in2 = 8  and y2>=16  then y2=y2-dy2
5820 if (in2 = 7  or in2 = 9 ) and y2>=16 then y2=y2-dy2
5840 if (in2 = 1  or in2 = 4  or in2 = 7 ) and x2<=496
     then x1=x1-dx1
5920 if (in1$="1" or in1$="3") and y1<=416 then y1=y1+dy1
5940 if (in1$="3" or in1$="6" or in1$="9") and x1<=200
     then x1=x1+dx1
6020 if (in2 = 1  or in2 = 3 ) and y2<=416 then y2=y2+dy2
6040 if (in2 = 3  or in2 = 6  or in2 = 9 ) and x2>=296
     then x2=x2+dx2
```

●リスト5

```
 150 int y3s,y3e
2040 line(256,16,256,495,0)
2050 y3s=rand() mod 200:y3e=y3s+(rand() mod 100)+100
2060 line(256,y3s,256,y3e,65535)
4250 if x>=235 and x<=275 and y>=y3s and y<=y3e then dx=-dx
```

ただこれをプレイするのは大変である。いい勝負をするには腕のたつ2人のプレイヤーが必要である。まあ仲のいい兄弟くらいであろう。

だいたいこのテのゲーム，最初は必ず2人のうちどちらかが勝ち続けることになっている。それは1人だけが熟練しているからである。2人で仲よくプレイするためには，できるだけ簡単なモードから慣らしていくのがよい。これが「お友達の法則」である。

話はそれるが，2人でとことんまで競ったゲームというのは案外思い出に残るものである。私の場合ずいぶん古い話だが「ハイパーオリンピック」は実によく覚えている。もうひとりは別に知り合いではなかったのだが，同じゲームセンターでレベルの高いスコアを出しているうちに，一緒の台でプレイするようになったのである。お互い名前も知らない仲だが，結構楽しんだ。あのゲームは，いかに速くボタンを押しつづけるかが勝負なのだが，2人のやり方はまったく違っていてその優劣を競ったのである。ゲームはライバルが大切である。

4) 改造その4　パドルが小さい

さて，次なる改造はさらにこのゲームを難しくしてくれる。パドルを小さくするのだ。いままで大雑把な感覚でやっていたのだが，より精密な動きを要求される。昔懐かしいブロック崩しを思い出すではないか。みんなこれで腕を上げていったものである。

これも改造は簡単である。リスト7が改造部分である。いままでパドルに5つ分のスプライトを用いていた処理を変数NPに変えたのである。これによって玉の衝突判定も必然的に範囲が変わる。

小さいパドルでのプレイは非常な集中力を必要とする。玉の行方を必死で追わなくてはならない。細かいキー操作も必要である。このような状況ではキーボード入力よりもジョイスティックのほうがいいかもしれない。キーボードのリピート機能はあまりにも貧弱だから，きっと操作ミスをしてしまうだろう。X-BASICの課題である。

さて，今日はいろいろな改造を試みたが，意外に変更点が少なかったことはリストを見てもらえばわかるであろう。このようにあとから機能を追加していくやり方は，非常に楽なのである。それにはまず基本形として簡単なプログラムを組み，あとはアイデアを凝らす→追加，修正する，といった手順である。毎日少しずつこれをやっているうちに，すごいゲームができるかもしれない。

5) 改造その5　スピードの変化

しかし，このゲームを作ってみると，一生懸命やっても，いろいろと工夫しないと案外スピードが遅いことがわかった。とはいうものの，いまは使うことを拒否しているが，バックにはまだコンパイラという強い味方がついている。このゲームもコンパイルしたらもっとゆっくりする方向で修正をしないと，とてもやってられないスピードになるかもしれない。

さてしかし，インタプリタのままで，このうえさらに玉のスピードに変化をつけてみたくなった。ただ速くしたり遅くしたりするだけなら，変数dxやdyなどの値を大きくしたり小さくしたりしてやればよい。しかし，ここではプレイの最中に玉のスピードが変化するようにした（リスト8）。

このプログラム，dxやdyをむやみに大きくしたり小さくしたりすると衝突判定が難しくなってしまう。なにせ座標がお互い飛

●リスト6

```
 130 int a,b,da,db
1150 sp_def(11,SP1)
2450 da=-dx
2550 a=255 :b=215        :db=16
3150 if b<=16 then db=-db
3250 if b>=480 then db=-db
3251 /*if a<=16 then da=-da:goto 3264
3252 if a<=16 then {
3253 }
3254 if a>=x1-16 and a<=x1+16 then {
3255    if b> y1-24 and b<y1      then da=16 :db=-32
3256    if b>=y1    and b<y1+16   then da=abs(da):db=-24
3257    if b>=y1+16 and b<y1+32   then da=abs(da):db=-16
3258    if b>=y1+32 and b<y1+48   then da=abs(da):db=0
3259    if b>=y1+48 and b<y1+64   then da=abs(da):db=16
3260    if b>=y1+64 and b<y1+80   then da=abs(da):db=24
3261    if b>=y1+80 and b<y1+104  then da=16 :db=32
3262 }
3263 if a>=235 and a<=275 and b>=y3s and b<=y3e then da=-da
3264 if a>=480 then da=-da:goto 3300
3265 if a>=480 then {
3266 }
3267 if a>=x2 and a<=x2+32 then {
3268    if b> y2-24 and b<y2      then da=-16:db=-32
3269    if b>=y2    and b<y2+16   then da=-abs(da):db=-24
3270    if b>=y2+16 and b<y2+32   then da=-abs(da):db=-16
3271    if b>=y2+32 and b<y2+48   then da=-abs(da):db=0
3272    if b>=y2+48 and b<y2+64   then da=-abs(da):db=16
3273    if b>=y2+64 and b<y2+80   then da=-abs(da):db=24
3274    if b>=y2+80 and b<y2+104  then da=-16:db=32
3275 }
6250 a=a+da
6350 b=b+db
6650 if a>=0 and b>=0 then sp_move(11,a,b,11)
6700 until x<0 or x>496 or a<0 or a>496
6950 if a<0   then sp_move(11,0,b,11)   :ip2=ip2+1
7050 if a>496 then sp_move(11,496,b,11):ip1=ip1+1
```

●リスト7

```
 170 int np=3
3256    if b>=y1           and b<y1+16*np/5  then da=abs(da):db=-24
3257    if b>=y1+16*np/5 and b<y1+32*np/5  then da=abs(da):db=-16
3258    if b>=y1+32*np/5 and b<y1+48*np/5  then da=abs(da):db=0
3259    if b>=y1+48*np/5 and b<y1+64*np/5  then da=abs(da):db=16
3260    if b>=y1+64*np/5 and b<y1+80*np/5  then da=abs(da):db=24
3261    if b>=y1+80*np/5 and b<y1+80*np/5+24 then da=16 :db=32
3269    if b>=y2    and b<y2+16*np/5  then da=-abs(da):db=-24
3270    if b>=y2+16*np/5 and b<y2+32*np/5  then da=-abs(da):db=-16
3271    if b>=y2+32*np/5 and b<y2+48*np/5  then da=-abs(da):db=0
3272    if b>=y2+48*np/5 and b<y2+64*np/5  then da=-abs(da):db=16
3273    if b>=y2+64*np/5 and b<y2+80*np/5  then da=-abs(da):db=24
3274    if b>=y2+80*np/5 and b<y2+80*np/5+24 then da=-16:db=32
3600    if y>=y1    and y<y1+16*np/5  then dx=abs(dx):dy=-24
3700    if y>=y1+16*np/5 and y<y1+32*np/5  then dx=abs(dx):dy=-16
3800    if y>=y1+32*np/5 and y<y1+48*np/5  then dx=abs(dx):dy=0
3900    if y>=y1+48*np/5 and y<y1+64*np/5  then dx=abs(dx):dy=16
4000    if y>=y1+64*np/5 and y<y1+80*np/5  then dx=abs(dx):dy=24
4100    if y>=y1+80*np/5 and y<y1+80*np/5+24 then dx=16 :dy=32
4600    if y>=y2    and y<y2+16*np/5  then dx=-abs(dx):dy=-24
4700    if y>=y2+16*np/5 and y<y2+32*np/5  then dx=-abs(dx):dy=-16
4800    if y>=y2+32*np/5 and y<y2+48*np/5  then dx=-abs(dx):dy=0
4900    if y>=y2+48*np/5 and y<y2+64*np/5  then dx=-abs(dx):dy=16
5000    if y>=y2+64*np/5 and y<y2+80*np/5  then dx=-abs(dx):dy=24
5100    if y>=y2+80*np/5 and y<y2+80*np/5+24 then dx=-16:dy=32
5900 if in1$="2" and y1<=496-np*16 then y1=y1+dy1
6000 if in2 = 2  and y2<=496-np*16 then y2=y2+dy2
6400 for i=1 to np:sp_move(i,x1,y1-16+16*i,i):next
6500 for i=1 to np:sp_move(i+5,x2,y2-16+16*i,i+5):next
```

び飛びに変わっていくので，その軌跡が交わったかどうか，よくわからなくなってしまうのである。dx，dyの値はこの点で細心の注意が必要となる。こういう値の設定はあまり考えすぎると疲れてしまうので，ある程度見当をつけたら少しずつ値を変えて試してみることが必要である。こういうときは2人でプレイできないので，片方のカベを完全に反射させてしまうのがよい。リスト3の3300行と4300行はコメントになっているが，このデバッグ方法のための処理である。

6）　改造その6　変化球

その昔，星飛雄馬は球質の軽さから直球に不安を感じ，大投手金田に変化球を教えてくれと頼んだ。そこで金田は「若いうちからそんなことでどうする，大リーグもびっくりするような新しい球を考えるんや」と一喝したという。かくして大リーグボールは生まれてくるわけだが，それにあやかって変化球を取り入れてみようではないか。

変化球の処理は思ったほど難しくない。要は，何もないところで玉の速さdx，dyを別々に変えてやれば曲がるのである。問題は「連続してぐぐーっと曲げる」か「ある場所でカクッと曲げる」かの選択である。でも後者はそこに壁があるのと似たようになってしまうので，ここでは連続して曲げていくことにしよう。それにはループが回るごとに，少しずつdxやdyを変化させてやればよいのだ（リスト9）。

これをプレイしてみると玉が逃げるように動いたり，逆に向かってきたりして，いままでとはまったく違った感じのゲームになる。曲がり方がまったく同じだと飽きてしまうので乱数を用いた。ゲームに乱数はつきものであるから使い方は覚えておいたほうがよい。この改造も玉とパドルの衝突判定がネックになってくる。

●リスト8

```
 120 int ic
2150 ic=0
2950 if ic mod 100=50 then dx=2*dx:da=2*da
2970 if ic mod 100=70 then dx=dx/2:da=da/2
3254 if a>=x1-abs(da) and a<=x1+abs(da) then {
3267 if a>=x2 and a<=x2+abs(da)*2 then {
3450 if x>=x1-abs(dx) and x<=x1+abs(dx) then {
4450 if x>=x2 and x<=x2+abs(dx)*2 then {
6670 ic=ic+1
6680 /*if (x>480 and dx>0) then dx=-dx
6685 /*if (a>480 and da>0) then da=-da
6690 /*if (x<16  and dx<0) then dx=-dx
6695 /*if (a<16  and da<0) then da=-da
6700 until (x<0 and dx<0) or (x>496 and dx>0) or (a<0 and da<0)
or (a>496 and da>0)
```

●リスト9

```
2960 if ic mod 100>=80 and ic mod 100<=89 then dy=dy+2:db=db+2
2980 if ic mod 100>=90 and ic mod 100<=99 then dy=dy-2:db=db-2
```

●リスト10

```
 350 dim char SP2(255)
 550 console ,,0
 750 for i=0 to 15:for j=0 to 15
 760    SP1(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/2+2
 765    SP2(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/2+8
 770 next:next
1130 for i=1 to 10:sp_def(i,SP2):next
1250 for i=2 to 7:sp_color(i,rgb(38-i*4,0,0)):next
1260 for i=2 to 7:sp_color(i+6,rgb(0,38-i*4,0)):next
1450 ic1=rgb(10,20,30)
2070 line(256,y3s,256,y3e,40000)
```

●リスト11

```
 140 int dao,dxo
 560 m_init():for i=1 to 6:m_alloc(i,200):m_assign(i,i):next
 570 m_trk(1,"@1132|:256ceabegfagfedcegbcegbcegbcegb:|")
 571 m_trk(2,"@50o3go4|:256c8.c16c.e8d8c8g8.g16g.g8f8e8d8.d16d.f8e8
d8c8o5c8o4b8a8g8f8e8d8:|")
 572 m_trk(3,"@1132fefef")
 573 m_trk(4,"@10164gbgbgbgb")
 574 m_trk(5,"@15cc8.c16ce-8.d16d8.c16c8.o3b16o4c2")
 575 m_trk(6,"@15g2o5ed8g8e8d8c8o4a8g2a8o5c8c8d8r8o4c8d8c16d16e1")
 580 m_play(1)
2080 m_stop(1)
2120 m_play(2)
3100 if y<=16 then dy=-dy:m_play(3)
3150 if b<=16 then db=-db:m_play(3)
3200 if y>=480 then dy=-dy:m_play(3)
3210 dao=da:dxo=dx
3250 if b>=480 then db=-db:m_play(3)
6696 if da<>dao then m_play(4)
6697 if dxo<>dx then m_play(4)
6750 m_stop(2):m_play(5)
7500 locate 25,30:print "                    ";
8150 m_play(6)
8350 locate 20,1:print "                                        ";
8400 locate 22,30:print "                                   ";
```

3.改造のつぎは修飾

以上6つの改造を試みたが，プログラムは改造ばかりが能ではない。もうひとつの大切なものが仕上げのデコレーション，修飾である。今回は2通りの修飾を考える。

1）　修飾その1　色を使う

いままでのリストでは味気のない真四角の玉と長方形のパドルが動いているという単調な画面であった。X68000の65536色を生かすこともせず，使った色もたったの2色（白と黒）である。せっかくスプライトは色などに関係なくスピードを保てるのだから，そろそろ使ってみたい。

このための修正リストがリスト10である。この程度の変更で画面の感じもグッと違ってくる。残念ながらセンスのない私にはこの程度しかできないが，センスのある（二科展に入選するような）人がやればきっと魅力的な画面になるだろうと思う。

2）　修正その2　音をつける

なんといったって効果音のないゲームは寂しい。これはほかの機能をほとんど考えずにつけ加えることができるから楽である。効果音はFM音源やAD PCMを使えばよい。AD PCMはデータを作らなくてはならないので，ここではFM音源を使おう。

改造例がリスト11である。ちょっとスピードは気になるが，自分の好きな音楽を用いれば自然とよい気持でプレイができようというものである。BGMを流す（これもFM音源で）という手もある。最も簡単で最も効果のある修飾である。

以上，すべての機能を統合してできたのが最後のリスト12である。今日のテーマは「スプライト」であろうか「後生説」であろうか。まあ，ゲームは簡単なものから始め，改造をし，最後に修飾すればそんなに苦労なく進んでいけるものなのである。

せっかくのX68000なのだからスプライトやFM音源やカラーを使って面白いプログラムに変えていこうではないか。

●リスト12

```
 100 int ai,x,y,dx,dy,in1,in2,ip1,ip2,ipmax=5
 120 int ic
 130 int a,b,da,db
 140 int dao,dxo
 150 int y3s,y3e
 170 int np=3
 200 str in1$,in2$
 300 dim char SP1(255)
 350 dim char SP2(255)
 400 /* initialize
 500 screen 1,3,1,1
 550 console ,,0
 560 m_init():for i=1 to 6:m_alloc(i,200):m_assign(i,i):next
 570 m_trk(1,"@1132|:256ceabegfagfedcegbcegbcegbcegb:|")
 571 m_trk(2,"@50o3go4|:256c8.c16c.e8d8c8g8.g16g.g8f8e8d8.d16d.
f8e8d8c8o5c8o4b8a8g8f8e8d8:|")
 572 m_trk(3,"@1132fefef")
 573 m_trk(4,"@10164gbgbgbgb")
 574 m_trk(5,"@15cc8.c16ce-8.d16d8.c16c8.o3b16o4c2")
 575 m_trk(6,"@15g2o5ed8g8e8d8c8o4a8g2a8o5c8c8d8r8o4c8d8c16d16e
1")
 580 m_play(1)
 600 /* sprite pattern definition
 700 for i=0 to 255:SP1(i)=1:next
 750 for i=0 to 15:for j=0 to 15
 760     SP1(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/2+2
 765     SP2(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/2+8
 770 next:next
 800 sp_clr(0,255)
 900 sp_off(0,127)
 1000 sp_disp(1)
 1100 for i=0 to 10:sp_def(i,SP1):next
 1130 for i=1 to 10:sp_def(i,SP2):next
 1150 sp_def(11,SP1)
 1200 sp_color(1,rgb(30,30,30))
 1250 for i=2 to 7:sp_color(i,rgb(38-i*4,0,0)):next
 1260 for i=2 to 7:sp_color(i+6,rgb(0,38-i*4,0)):next
 1300 /* draw wall
 1400 ic1=rgb(30,30,30)
 1450 ic1=rgb(10,20,30)
 1500 box(0,0,511,15,ic1,&HFFFF)
 1600 paint(256,8,ic1)
 1700 box(0,496,511,511,ic1,&HFFFF)
 1800 paint(256,504,ic1)
 1900 repeat
 2000 ip1=0:ip2=0
 2040 line(256,16,256,495,0)
 2050 y3s=rand() mod 200:y3e=y3s+(rand() mod 100)+100
 2060 line(256,y3s,256,y3e,65535)
 2070 line(256,y3s,256,y3e,40000)
 2080 m_stop(1)
 2100 repeat
 2120 m_play(2)
 2150 ic=0
 2200 locate 28,1:print ip1;:locate 31,1:print "-";:print ip2
 2300 /* initial position & velocity
 2400 if x<0 then dx=24 else dx=-24
 2450 da=-dx
 2500 x=255 :y=215         :dy=16
 2550 a=255 :b=215         :db=16
 2600 x1=0  :y1=215:dx1=16:dy1=24
 2700 x2=496:y2=215:dx2=16:dy2=24
 2800 /* start
 2900 repeat
 2950 if ic mod 100=50 then dx=2*dx:da=2*da
 2960 if ic mod 100>=80 and ic mod 100<=89 then dy=dy+2:db=db+2
 2970 if ic mod 100=70 then dx=dx/2:da=da/2
 2980 if ic mod 100>=90 and ic mod 100<=99 then dy=dy-2:db=db-2
 3000 /* boundary condition
 3100 if y<=16 then dy=-dy:m_play(3)
 3150 if b<=16 then db=-db:m_play(3)
 3200 if y>=480 then dy=-dy:m_play(3)
 3210 dao=da:dxo=dx
 3250 if b>=480 then db=-db:m_play(3)
 3251 /*if a<=16 then da=-da:goto 3264
 3252 if a<=16 then {
 3253 }
 3254 if a>=x1-abs(da) and a<=x1+abs(da) then {
 3255    if b> y1-24       and b<y1       then da=16 :db=-32
 3256    if b>=y1          and b<y1+16*np/5  then da=abs(da):db=-
24
 3257    if b>=y1+16*np/5 and b<y1+32*np/5  then da=abs(da):db=-
16
 3258    if b>=y1+32*np/5 and b<y1+48*np/5  then da=abs(da):db=0
 3259    if b>=y1+48*np/5 and b<y1+64*np/5  then da=abs(da):db=1
6
 3260    if b>=y1+64*np/5 and b<y1+80*np/5  then da=abs(da):db=2
4
 3261    if b>=y1+80*np/5 and b<y1+80*np/5+24 then da=16 :db=32
 3262 }
 3263 if a>=235 and a<=275 and b>=y3s and b<=y3e then da=-da
 3264 /*if a>=480 then da=-da:goto 3300
 3265 if a>=480 then {
 3266 }
 3267 if a>=x2 and a<=x2+abs(da)*2 then {
 3268    if b> y2-24 and b<y2      then da=-16:db=-32
 3269    if b>=y2      and b<y2+16*np/5  then da=-abs(da):db=-24
 3270    if b>=y2+16*np/5 and b<y2+32*np/5  then da=-abs(da):db=
-16
 3271    if b>=y2+32*np/5 and b<y2+48*np/5  then da=-abs(da):db=
0
 3272    if b>=y2+48*np/5 and b<y2+64*np/5  then da=-abs(da):db=
16
 3273    if b>=y2+64*np/5 and b<y2+80*np/5  then da=-abs(da):db=
24
 3274    if b>=y2+80*np/5 and b<y2+80*np/5+24 then da=-16:db=32
 3275 }
 3300 /*if x<=16 then dx=-dx:goto 4300
 3400 if x<=16 then {
 3410 }
 3450 if x>=x1-abs(dx) and x<=x1+abs(dx) then {
 3500    if y> y1-24 and y<y1      then dx=16 :dy=-32
 3600    if y>=y1      and y<y1+16*np/5  then dx=abs(dx):dy=-24
 3700    if y>=y1+16*np/5 and y<y1+32*np/5  then dx=abs(dx):dy=-
16
 3800    if y>=y1+32*np/5 and y<y1+48*np/5  then dx=abs(dx):dy=0
 3900    if y>=y1+48*np/5 and y<y1+64*np/5  then dx=abs(dx):dy=1
6
 4000    if y>=y1+64*np/5 and y<y1+80*np/5  then dx=abs(dx):dy=2
4
 4100    if y>=y1+80*np/5 and y<y1+80*np/5+24 then dx=16 :dy=32
 4200 }
 4250 if x>=235 and x<=275 and y>=y3s and y<=y3e then dx=-dx
 4300 /*if x>=480 then dx=-dx:goto 5300
 4400 if x>=480 then {
 4410 }
 4450 if x>=x2 and x<=x2+abs(dx)*2 then {
 4500    if y> y2-24 and y<y2      then dx=-16:dy=-32
 4600    if y>=y2      and y<y2+16*np/5  then dx=-abs(dx):dy=-24
 4700    if y>=y2+16*np/5 and y<y2+32*np/5  then dx=-abs(dx):dy=
-16
 4800    if y>=y2+32*np/5 and y<y2+48*np/5  then dx=-abs(dx):dy=
0
 4900    if y>=y2+48*np/5 and y<y2+64*np/5  then dx=-abs(dx):dy=
16
 5000    if y>=y2+64*np/5 and y<y2+80*np/5  then dx=-abs(dx):dy=
24
 5100    if y>=y2+80*np/5 and y<y2+80*np/5+24 then dx=-16:dy=32
 5200 }
 5300 /*key or joystick sense
 5400 in1$=inkey$(0)
 5500 /*in1 =stick(2)
 5600 in2 =stick(1)
 5700 if in1$="8" and y1>=16 then y1=y1-dy1
 5720 if (in1$="7" or in1$="9") and y1>=16 then y1=y1-dy1
 5740 if (in1$="1" or in1$="4" or in1$="7") and x1>=15
      then x1=x1-dx1
 5800 if in2 = 8  and y2>=16  then y2=y2-dy2
 5820 if (in2 = 7  or in2 = 9 ) and y2>=16 then y2=y2-dy2
 5840 if (in2 = 1  or in2 = 4  or in2 = 7 ) and x2<=496
      then x1=x1-dx1
 5900 if in1$="2" and y1<=496-np*16 then y1=y1+dy1
 5920 if (in1$="1" or in1$="3") and y1<=416 then y1=y1+dy1
 5940 if (in1$="3" or in1$="6" or in1$="9") and x1<=200
      then x1=x1+dx1
 6000 if in2 = 2  and y2<=496-np*16 then y2=y2+dy2
 6020 if (in2 = 1  or in2 = 3 ) and y2<=416 then y2=y2+dy2
 6040 if (in2 = 3  or in2 = 6  or in2 = 9 ) and x2>=296
      then x2=x2+dx2
 6100 /*movement of ball and pads
 6200 x=x+dx
 6250 a=a+da
 6300 y=y+dy
 6350 b=b+db
 6400 for i=1 to np:sp_move(i,x1,y1-16+16*i,i):next
 6500 for i=1 to np:sp_move(i+5,x2,y2-16+16*i,i+5):next
 6600 if x>=0 and y>=0 then sp_move(0,x,y,0)
 6650 if a>=0 and b>=0 then sp_move(11,a,b,11)
 6670 ic=ic+1
 6680 /*if (x>480 and dx>0) then dx=-dx
 6685 /*if (a>480 and da>0) then da=-da
 6690 /*if (x<16  and dx<0) then dx=-dx
 6695 /*if (a<16  and da<0) then da=-da
 6696 if da<>dao then m_play(4)
 6697 if dxo<>dx then m_play(4)
 6700 until (x<0 and dx<0) or (x>496 and dx>0) or (a<0 and da<0)
or (a>496 and da>0)
 6750 m_stop(2):m_play(5)
 6800 /*point incriment and display
 6900 if x<0   then sp_move(0,0,y,0)  :ip2=ip2+1
 6950 if a<0   then sp_move(11,0,b,11)  :ip2=ip2+1
 7000 if x>496 then sp_move(0,496,y,0):ip1=ip1+1
 7050 if a>496 then sp_move(11,496,b,11):ip1=ip1+1
 7100 locate 28,1:print ip1;:locate 31,1:print "-";:print ip2
 7200 repeat
 7300 locate 25,30:print "(c)ontinue";:in1$=inkey$
 7400 until in1$="c"
 7500 locate 25,30:print "          ";
 7600 until ip1>=ipmax or ip2>=ipmax
 7700 /*display winner
 7800 locate 20,1
 7900 if ip1>ip2 then print "won by player 1(";ip1;"-";ip2;")"
 8000 if ip1<ip2 then print "won by player 2(";ip1;"-";ip2;")"
 8100 repeat
 8150 m_play(6)
 8200 locate 22,30:print "(c)ontinue or e(x)it";:in1$=inkey$
 8300 until in1$="c" or in1$="x"
 8350 locate 20,1:print "                              ";
 8400 locate 22,30:print "                    ";
 8500 until in1$="x"
 8600 screen 1,3,1,1
 8700 end
```

PURE PASCAL

PASCALプログラミングへの招待〈6〉

実行時チェック・Cとのリンク

PASCAL入門の連載もいよいよ大詰め，今回は“高級言語”PASCALの一面でもある実行時のエラーチェックとより高度な使用法のためのC言語とのリンク，コンパイルオプションなどについて解説します。

Fujiki Takeshi / Fujii Yoshimi
藤木健士/藤井義巳

1 実行時チェック

実行時チェックというのは実際に処理を行ったときに発生するエラーのチェックのことです。

たとえば，

```
var    i: 1..10;
```

と宣言した場合，変数iには1より小さい値や10より大きな値をとることができません。このような'変数の範囲を越える'といった種類のエラーは文法エラーと異なりコンパイル時には検出することができません。実際に処理を実行して，変数に値が代入される時点になってはじめてその値がわかるからです。では，どのようにしてこのチェックを行うのでしょうか？　それは，コンパイルするときに実行時エラーが発生する可能性のある部分にはチェックルーチンを組み込んでしまうのです。

それでは例をとって説明することにしましょう。リスト1を見てください。このプログラムはいくつかの数字を入力してその数字の中の最大のものを表示するプログラムです。このプログラムでは最大5個の数字から最大値を求めることができます。これをコンパイルして実行すると入力要求を出してくるので数字を次のように入れてみましょう。

```
123 53 36 652 0
```

すると次のように最大値を返してくるでしょう

```
最大の数値        652
```

それでは今度は入力要求のあとに次のように入力してみましょう。

```
431 3431 23 43 79 867 234 0
```

すると今度は実行時エラーが発生して次のように表示されるでしょう。

```
実行時エラー        LINE 18
スカラーまたは部分範囲が範囲を越えています。
```

これは18行目のread(a[i])を実行しようとした際にiが6を越えてしまったために起こったのです。これは，配列aは整数6つ分の領域しか宣言していないためにこのような結果になるのです。

もしこの確認を行わずに実行を続けてしまうとa[7]に数値を入力してしまうことになり，確保していない記憶領域の更新をしてしまいます。この実行時チェックの機能がなければ，別の変数の値がいつのまにか変わっていたり，悪いとき（良いとき？）にはアドレスエラーなどでプログラムが止まってしまうこともあるでしょう。この手のバグを探すのは非常に大変です（経験したことはありませんか？）。

実行時に発生するエラーはほかに，0で割ろうとしたり，オーバーフローが発生したり，メモリが足りなくなったりしたときにも発生します。

一般にPASCALで書かれたプログラムはこのように実行時のエラーの検出を行います。Cでは実行時のエラーのチェックはやりませんので，そのようなチェックを行いたいときにはプログラマがチェックの部分をコーディングする必要があります。しかし，PASCALでは実行時エラーが発生する可能性があるすべての部分で実行時チェックの処理を行うので処理する時間がCで書かれたプログラムよりも長くなり，実行ファイルのサイズもチェックルーチンの分だけ大きくなります。

リスト1

```
 1:     program MaxNumber(input,output);
 2:
 3:     const   TableSize = 6;
 4:
 5:     var     i:integer;
 6:             a:array[1..TableSize] of 0..MAXINT;
 7:             max:integer;
 8:
 9:     begin
10:
11:             for i:=1 to TableSize do
12:                     a[i] := 0;
13:
14:             i:=0;
15:             writeln('数字を入力して下さい。最後の数字の後に
0を入力してください');
16:             repeat
17:                     i := i + 1;
18:                     read(a[i]);
19:             until a[i] = 0;
20:
21:             max := 0;
22:             for i:= 1 to TableSize do
23:                     if max < a[i] then
24:                     max := a[i];
25:
26:             writeln('最大の数値',max)
27:     end.
```

2 Cとのリンク

前回はPASCALのプログラム中にアセンブリ言語で書いた処理を書いてそれを実行する方法を紹介しました。今月はそれに続いて今回はCのライブラリ関数を外部関数としてPASCALのプログラムから呼び出す方法を紹介します。リスト2を見てください。これはファイルを行単位でソートするプログラムです。29行から33行目までで外部関数の宣言を行っています。
外部関数の宣言はextern指令を使って次のように書きます。

```
FUNCTION 関数名(パラメータリスト) :戻り値;
extern;
PROCEDURE 手続き名 (パラメータリスト) ;
extern;
```

リスト2のサンプルプログラムではファイル入出力関係のCのライブラリ関数を宣言しています。Cでは大文字と小文字の区別をするのでPurePASCALでも外部関数を宣言するときには大文字と小文字を区別するようにしています。注意してください。ここでもうひとつ注意すべきことがあります。30行目を見てください。

```
function fgets(fp:FILEP; n:integer; var s:
STR255):STRP;extern;
```

となっていますね。ところがCのライブラリマニュアルを見るとfgetsの引数について次のように書いてあります。

```
char    *fgets(string, n, stream);
char    *string;  /* 読み込んだ文字列の格
                     納領域へのポインタ  */
int      n;        /*  読み込む文字数  */
FILE    *stream;  /*  FILE構造体へのポ
                     インタ    */
```

引数の順序が逆になっていますね。これはPASCALでは引数がCとは逆に積む(11月号を参考にしてください)からです。

それからファイル構造体へのポインタの型ですがFILEPつまり^CHARというふうにPASCALのプログラム上では文字へのポインタというふうに記述していますが，これは4バイトの領域をとる型であればなんでもかまいません。ただfopenで得た値をほかのファイル入出力関数に渡したいだけなのでこれで十分なのです。

もしPASCALのプログラムの中でファイル構造体の内容を参照したいなら，

```
FILEP = ^CHAR
```

の代わりに次のように書けば一応アクセスできるのではないかと思われます。ただしPurePASCALでのアライメントがCと一致していないと思いますので，普通のやり方では完全に正しい値がくることを保証できません。使う必要のある場合はいろいろ試してみてデータが正しく受け渡されるかたちのレコード型を定義してください。

```
type
        FILESTRUCTURE = RECORD
                __ptr:^char;
                __cnt:integer;
                __base:^char;
                __flag:integer;
                __bsize:integer;
                __file:char;
                __pback:char
        end;
        FILEP =^FILESTRUCTURE;
```

これはCのインクルードファイルstdio.hに記述してある次の構造体と同じかたちです。

```
extern  FILE    {
        char    *__ptr;
        int     *__cnt;
        char    *__base;
        int     *__flag;
        int     *__bsize;
        char    *__file;
        char    *__pback;
        }       *__iob [__NFILE] ;
```

ここで，

```
type  FILE = record
```

というようにFILEという名前を使わないでください。FILEという名前は事前に宣言された型としてすでに存在します。それでここで上のように書いてしまうとreadやwriteでFILEに書き込むことができなくなってしまいます。注意してください。

あとは通常のPASCALのプログラムと同じです。それではコンパイルしてみましょう。いつものように次のようにコマンドを入力してください。

```
>pascal sort.pas
```

するとコンパイルとアセンブルを実行して終了します。外部関数を使っていないとき（extern指令を使っていないとき）はリンクまで実行しますが，外部関数を呼んでいるときにはコンパイラはどこに外部関数が存在するかわからないのでリンクは実行しません。それでマニュアルでリンクをする必要があります。今回呼んでいるファイル入出力関数はXCの場合CLIB.Aに入っていますの

で，次のようにコマンドを入力してください。

```
>LK sort.o paslib.a clib.a
```

CLIB.AやPASLIB.Aがカレントディレクトリにない場合はフルパスで指定してください。以上でSORT.Xという実行ファイルができあがります。それでは実行してみてください。

```
Input File name.:
```

と入力ファイル名を聞いてきますのでファイル名を入力してください。そのあと次のように出力ファイル名を聞いてきます。

```
Output file name.:
```

なにも入力せずにリターンキーを打つとコンソールに結果が出力されます。ファイルに出力したいときはファイル名を入力してリターンキーを押してください。ファイルの内容が並べ替えられているでしょう！

ちなみに，このプログラムではソートのアルゴリズムとしては総合的にもっとも速いといわれているクイックソートを使っています。簡単にアルゴリズムを説明すると，どれか要素をひとつを取ってきてそれ以外のものをそれより大きいグループと小さいグループに分けて，その分けたグループの中でも同じ動作を行い，それより大きなグループと小さなグループに分ける。これを続けてグループの中が1になったらソートを終了する。こんな感じです。

このアルゴリズムは再帰を使って書くと簡単です。サンプルプログラムでもそのようになっています。興味のある人はプログラムを解析してみてください。ソートのアルゴリズムにはこのほかにも，いろんな方法がありますが総合的にクイックソートより高速なものは考え出されていません。もしこれより速いアルゴリズムを考えたらすごいですよ！　ひと儲けできるでしょう。

話がそれてしまいましたが，この外部関数をリンクする方法を使うと，Cで提供されている豊富な関数を用いることができます。これで，プログラムの幅が非常に広くなったのではないでしょうか？

3 PurePASCALのコンパイルオプション

PurePASCALはいくつかのコンパイルオプションを使用することができます。それではそれらについてひとつずつ説明していきます。

ファイルインクルード

これはCにおける#includeと同じ機能を提供するもので，指定されたファイルをこの指令が書かれた位置に挿入します。

書式

{$Ifilename}

例

{$Igraphic.inc}

このように書くとgraphic.incというファイルをこの指令が書かれた位置に展開します。

インラインアセンブラ

{$asm}　から {$endasm} までのあいだにアセンブリ言語のプログラムを埋め込みます。IOCSコールやDOSを使いたいときなどに使ってください。

書式

{$asm}

　　アセンブリ言語のプログラム

{$endasm}

インラインアセンブラに関してはOh!Xの11月号に詳しく説明してあります。参考にしてください。

スタック領域のサイズの指定

実行ファイルのスタック領域を1Kバイト単位で指定します。省略値は64Kバイトです。プログラム中で大きな配列などを使用したり，かなり深い再帰を行うときには大きめに設定してください。これによりスタックオーバーフローを回避することができます。

書式

{$Ssize}

例

{$S256}

256Kバイトのスタック領域を確保します。

ヒープ領域のサイズの指定

実行ファイルのヒープ領域を1Kバイト単位で指定します。省略値は64Kバイトです。NEW手続きで確保する領域が大きくなる可能性があるときには大きな値を指定してください。

書式

{$Hsize}

例

{$H256}

256Kバイトのヒープ領域を確保します。

実行時チェックのコード生成スイッチ

今回説明したようにPASCALは実行時にいろいろなエラーチェックを行います。そのために実行速度がある程度低下しますし，プログラムサイズも大きくなります。{$nocheck}を指定すると，そこから先の部分では実行時チェックのコードが生成されません。{$check}を指定すると，そこから再び実行時チェックが再開されます。コンパイル開始時には {$check} の状態になっていま

す。プログラム作成時には {$check} を行い，デバッグが済んだら {$nocheck} にしてスピードアップするとよいでしょう。

書式

{$check}

{$nocheck}

＊　　＊　　＊

11月号で説明したアセンブリ言語とのリンク，そしてCとのリンクを使えばかなりのことができると思います。いろんなプログラムを作ってみてください。

リスト2

```
  1:   (* sort.pas *)
  2:   (*
  3:      ファイルを行単位でソートする。
  4:      ファイルの入出力にはCの標準関数を利用する。
  5:      ヒープを使うので、大きなファイルをソートする時は、
  6:      コンパイラ指令でヒープサイズを大きくするとよい。
  7:         Oct.1990 by Chack'n
  8:   *)
  9:
 10:   program sort(input, output);
 11:   const
 12:      MAXLINES = 10000;   (* 最大1万行まで *)
 13:   type
 14:      (* テキストファイルの1行の文字数は255文字まで *)
 15:      STR255 = packed array[1..255]of char;
 16:      STRP = ^STR255;
 17:      FILEP = ^CHAR;      (* コンパイラをだましている *)
 18:      (* ソートのための配列 *)
 19:      STRING_ARRAY = array[1..MAXLINES]of STRP;
 20:   var
 21:      ifn, ofn:STR255;
 22:      TextLines:STRING_ARRAY;
 23:      lines, n:integer;
 24:   (*
 25:      リンクするCの関数のEXTERN宣言。
 26:      関数名の大文字と小文字は区別される。
 27:      戻り値が不用な時はprocedureとして宣言すればよい。
 28:   *)
 29:      function  fopen(var ftype:STR255; var fname:STR255):FI
LEP;extern;
 30:      function  fgets(fp:FILEP; n:integer; var s:STR255):STR
P;extern;
 31:      procedure fputs(fp:FILEP; var s:STR255);extern;
 32:      procedure fclose(fp:FILEP);extern;
 33:      procedure putchar(c:integer);extern;
 34:
 35:   (*
 36:      ここから先はPASCALのプログラム
 37:   *)
 38:      (* C形式の文字列を標準入力から入力する。 *)
 39:      function getline(var line:STR255):integer;
 40:      var
 41:         i:integer;
 42:         c:char;
 43:      begin
 44:         i := 1;
 45:         repeat
 46:            read(c);
 47:            if not eoln then
 48:               begin
 49:              line[i] := c;
 50:              i := i + 1
 51:               end
 52:         until eoln or (255 <= i);
 53:         line[i] := chr(0);
 54:         getline := i - 1;
 55:         readln
 56:      end;
 57:
 58:      (* テキストファイルをまるごと読み込む *)
 59:      function readfile(var fn:STR255):integer;
 60:      var
 61:         i, j:integer;
 62:         r:STRP;
 63:         ftype, s:STR255;
 64:         fp:FILEP;
 65:      begin
 66:         ftype[1] := 'r'; ftype[2] := Chr(0);
 67:         fp := fopen(ftype, fn);
 68:         if fp = Nil then
 69:             begin
 70:            writeln('file open error. <',ifn,'>');
 71:            readfile := 0
 72:             end
 73:         else
 74:             begin
 75:            i := 1;
 76:            repeat
 77:               r := fgets(fp, 255, s);
 78:               if r <> Nil then
 79:                   begin
 80:                  new(TextLines[i]);
 81:                  for j := 1 to 255 do
 82:                     TextLines[i]^[j] := s[j];
 83:                  i := i + 1
 84:                   end;
 85:            until (MAXLINES < i) or (r = Nil);
 86:            fclose(fp);
 87:            readfile := i - 1
 88:             end
 89:      end;
 90:
 91:      (* ファイルに出力 *)
 92:      procedure writefile(var fn:STR255; lines:integer);
 93:      var
 94:         i:integer;
 95:         ftype:STR255;
 96:         fp:FILEP;
 97:      begin
 98:         ftype[1] := 'w'; ftype[2] := Chr(0);
 99:         fp := fopen(ftype, fn);
100:         if fp = Nil then
101:            writeln('file open error. <',ifn,'>')
102:         else
103:             begin
104:            for i := 1 to lines do
105:                begin
106:               fputs(fp, TextLines[i]^);
107:               dispose(TextLines[i])
108:                end;
109:            fclose(fp)
110:             end
111:      end;
112:
113:      (* 標準出力に出力 *)
114:      procedure WriteToScreen(lines:integer);
115:      var
116:         i, j:integer;
117:      begin
118:         for i := 1 to lines do
119:             begin
120:            j := 1;
121:            while (TextLines[i]^[j] <> Chr(0)) and (j < 255)
 do
122:                begin
123:               putchar(Ord(TextLines[i]^[j]));
124:               j := j + 1
125:                end;
126:            dispose(TextLines[i])
127:             end;
128:      end;
129:
130:      (* クイックソート *)
131:      procedure qsort(l, r:integer);
132:      var
133:         i, m:integer;
134:
135:         (* 行を交換する *)
136:         procedure swap(var s1, s2:STRP);
137:         var
138:            st:STRP;
139:         begin
140:            st := s1;
141:            s1 := s2;
142:            s2 := st
143:         end;
144:
145:      begin { qsort }
146:         if l < r then
147:             begin
148:            m := l;
149:            for i := l + 1 to r do
150:                begin
151:               if TextLines[i]^ < TextLines[l]^ then
152:                   begin
153:                  m := m + 1;
154:                  swap(TextLines[i], TextLines[m]);
155:                   end;
156:                end;
157:            swap(TextLines[l], TextLines[m]);
158:            qsort(l, m - 1);
159:            qsort(m + 1, r);
160:             end { if }
161:      end; { qsort }
162:
163:   (* メインルーチン *)
164:   begin { MAIN }
165:      write('Input file name.:');
166:      n := getline(ifn);
167:   (* 出力ファイル名でリターンだけを入力すると標準出力になる
。 *)
168:      write('Output file name.:');
169:      n := getline(ofn);
170:      lines := readfile(ifn);
171:      if lines <> 0 then
172:          begin
173:         qsort(1, lines);
174:         if n = 0 then
175:            WriteToScreen(lines)
176:         else
177:            writefile(ofn, lines)
178:          end
179:   end.
```

カード型データベース(3)

Izumi Daisuke 泉 大介

BASIC

X-BASICプログラミング調理実習もいよいよ最終回となりました。カード型データベースの最後にテキストファイルを扱うためのツールを加えておきましょう。また，付録ディスクにはVS2.Xに対応したバージョンが収録されています。

前回，前々回の2回を使って，カード型データベースを制作しました。使い心地はどうでしょうか。今回は調理実習の最後として，カード型データベースの能力を拡張する簡単なツールを作ることにしましょう。また，ディスクに収めたcbase.xは連載で発表したものに若干の手を加えてありますので，リファレンスマニュアルを用意しました。参照してください。

データの出入り口

データベースはそれ単体で完結してしまうものではありません。データの入り口はキーボードだけであってはいけないのです。最も身近なデータの入り口はワープロです。日記やちょっとしたデータの記録などにもワープロを使っている方は少なくないだろうと思います。またワープロは，ディスプレイ以外のデータの出口としても有用です。多くのデータの中からさまざまな条件で検索をかけ，見つけ出したデータを文書に取り込んで体裁よく仕上げるという作業を考えてみるとよくわかると思います。

前回までに作成したデータベースは，こういった他のアプリケーションとのデータのやり取りに関してはまったく考えられていません。そこで今回は，データベースファイルをテキストファイルとして出力する，テキストファイルをデータベースファイルとしてデータベースに取り込む2つのツールを作ってみることにしましょう。

●テキストデータの形式

データベース，表集計，ワープロは，通常それぞれ独自の形式でディスクにデータを保存しています。このため，データベースのファイルをワープロで読み込んだり，ワープロの文書を表集計ソフトで読み込んで使うことはできません。

そこで，アプリケーション間で共通のデータ形式が考えられました。ただし，独自の形式に比べてディスク入出力の効率は落ちますので，独自形式の他に共通形式もサポートするという形で実現されるのが普通です。

最も有名な共通形式はCSV形式と呼ばれる方法です。表集計ソフトでいうなら，表の横方向のデータをカンマで区切って並べ，縦方向の各行は改行で区切ろうというものです。カード型データベースでいうなら，データとデータをカンマで区切り，カードとカードは改行で区切ることになります。ただし，共通であるはずのCSV形式も厳密な定義があるわけではなく，文字列をダブルクォートでくくるものがあったり，カンマの代わりにタブでデータを区切るものがあったりとまちまちです。データ共通化の道は，まだその途に就いたばかりだといえるでしょう。

今回採用したテキストデータ形式も，X-BASICで簡単に扱え，さらに実用になるスピードが得られるものという条件を無視するわけにいかず，CSVやSYLKなどの既存の数種類の共通フォーマットとは異なったものになっています。まず，カード上の各々のデータは改行によって区切られます。これはfreads関数で読み出せるようにという配慮です。そして，それぞれのカードは改頁と改行によって区切られています。freads関数で読み出した文字列が改頁（chr$(12)）だった場合には，カードはそこで終わりと判断できるわけです。このデータ形式は，標準ワープロであるWP.X，標準エディタであるED.Xでそのまま扱うことができます。

テキストデータの出力

まずはデータベースファイルをテキストファイルとして出力することから考えてみましょう。現在使用されている全カードを対象にしようかとも思ったのですが，cards配列に登録されているカードだけに止めることにしました。条件に合致したカードだけを出力してワープロなどで再加工できるようにという配慮からです。

プログラムはリスト1のtextOutput関数です。2つのパラメータを受け取っていますが，最初のnfpはテキストを出力するファイルのファイル番号，次のnはcards配列に入っているカードの総数です。テキストを出力するファイルのオープンは，あとで説明するtools関数で行っています。

textOutput関数は簡単で，cards(0)～cards(n)までのカードをreadCard関数（これはデータベース本体で使用

していくもの）で順に読み込んでいきます。データはData配列にセットされますから，Data(0) ～Data(30)をCR+LF付きで出力していくだけです。ひとつのカードを出力したら，改頁+CR+LFを出力してカードの区切りとし，次のカードにとりかかります。プログラムを見ればなにをやっているのかすぐにわかると思います。

テキストデータの入力

テキストデータを読み込むほうも，データベース本体で使用している関数を使えば簡単にケリがつきます。基本的な方針としては，Data配列にテキストデータを読み込み，それをwriteCard関数で出力すればOKということになります。

textInput関数を見てください。textOutput関数同様に，こちらもnfpをパラメータとして受け取ります。nfpは今度はテキストを読み込むファイルのファイル番号を表しています。読み込める最大のカード数は全カード数（100枚）から現在使用中のカード数（usingCards枚）を引けば求められます。先月はcommands関数中にusingCards変数をとっていましたが，textInput関数でもusingCardsを参照できるようにこれを大域変数にすることにします。具体的にはcommands関数からusingCardsの宣言を削除し，プログラム先頭の大域変数宣言にusingCardsを付け加えてください。

textInput関数は，続いてdbasep配列に－１をセットします。読み込むテキストファイルから新しいカードを作るのですから，dbasep配列を未使用の状態にするわけですね。同時にData配列にも“”をセットして未使用にします。

あとはnfpから１行ずつデータを読み込み，dataEntry配列を参照しながらデータをセットすべき行に読み込んだデータをセットする作業を繰り返します。改頁を読み込んだらカードはそこで終了。newCard関数で新しいカード番号をもらいwriteCard関数でカードを書き込みます。usingCardsは当然ひとつ増えますね。

Toolsメニューの追加

今回作成したテキスト入出力は，データベース操作中にいつでも使用できるようにしておくのが便利です。そこで，データベースメニューにToolsというメニューを追加しておきましょう。commands関数でメニューを表示しているところにToolsを書き加え，続くswitch文の中に，

```
case 't'
case 'T'
  tools( chosen )
  break
```

という選択肢を加えておきます。

さてtools関数ですが，これは，

OutputText，InputText >

というメッセージを表示する，commands関数と同じようなメニュープログラムになっています。パラメータとして渡されたchosenは，cards配列に何枚のカードが登録されているかを表し，これはtext Output関数で利用されます。例によってメニュー選択のキー入力はswitch文によって場合分けされ，それぞれの処理が行われます。追いかけてみてください。

ディスクに収めたcbase.xには，このテキスト入出力に加え，カード設計の変更をツールとして用意してあります。これでよしと思ったはずの設計も，データを入力していく途中で不都合を感じて設計し直したくなる場合があります。すでに入力したデータは無駄にしたくない。この要求に応えるものです。今回のリストに入れなかったのは，予想以上に大きくなってしまったからです。使い方はリファレンスマニュアルを見ていただくとして，同様の機能を持ったものを自分で作ってみるというのはいい勉強になるだろうと思います。

最後に

X-BASICの人気が徐々に盛り上がりつつあるようです。最初は既存のBASICとはまったく異なったBASICであり，しかも動作速度が遅いということでないがしろにされがちでしたが，C言語を使うより容易にプログラムが作れること，コンパイルすれば十分実用になる速度を得られることが認められてきたためでしょう。かく言う私も連載を始めるまでほとんど使ったことはありませんでした。それが連載を続けていくうちに，まずはX-BASICでプログラムを作ってみて，それからCで作り直すなり，コンパイルしてからC言語の段階で手を入れるというアプローチをとるようになってきました。

プログラムの動作を確かめるのにこれほど手軽な言語はありません。しかも，Yetのようなリアルタイム性を備

えたゲームから，カード型データベースという実用的なアプリケーションを作るほどの能力も備えています。これからもプログラミングの友としてX-BASICを使っていくことでしょう。X-BASICの魅力をお伝えする機会がえられたことに，そして連載につきあってくださった読者の皆さんに，心よりの感謝を捧げたいと思います。

リスト1　データベース用追加ツール

```
10000 /*
10010 /*  追加ツール
10020 /*
10030 func tools( chosen )
10040   str ch
10050   int nfp, i, flag
10060   flag = 1
10070   while flag
10080     locate 0,31
10090     print "OutputText, InputText > ";chr$(5);
10100     ch = inkey$
10110     switch asc( ch )
10120       case 27                         /* ESC
10130         flag = 0
10140         break
10150       case 'o'
10160       case 'O'
10170         line$ = getLine( "出力ファイル名：" )
10180         nfp = fopen( line$, "c" )
10190         textOutput( nfp, chosen )
10200         fclose( nfp )
10210         break
10220       case 'i'
10230       case 'I'
10240         line$ = getLine( "入力ファイル名：" )
10250         error off
10260         nfp = fopen( line$, "r" )
10270         error on
10280         if nfp = -1 then {
10290           locate 0,31
10300           print line$+"が存在しません。";chr$(5);
10310           beep
10320           ch = inkey$
10330         } else {
10340           textInput( nfp )
10350           fclose( nfp )
10360         }
10370         break
10380     endswitch
10390   endwhile
10400 endfunc
10410 /*
10420 /*  ファイルへのテキスト出力
10430 /*
10440 func textOutput( nfp, n )
10450   int i, j
10460   for i=0 to n-1
10470     readCard( cards( i ))
10480     for j=0 to 30
10490       if dataEntry( j ) <> 255 then {
10500         fwrites( Data(j)+chr$(13)+chr$(10), nfp )
10510       }
10520     next
10530     fwrites( chr$(12)+chr$(13)+chr$(10), nfp )
10540   next
10550 endfunc
10560 /*
10570 /*  ファイルからのテキスト入力
10580 /*
10590 func textInput( nfp )
10600   int i, j
10610   for i=usingCards to 99 /* cMax-1
10620     if feof( nfp ) then break
10630     cls
10640     for j=0 to 30
10650       dbasep( j ) = -1
10660       Data( j ) = ""
10670     next
10680     for j=0 to 31
10690       if j < 31 then {
10700         if dataEntry( j ) = 255 then {
10710           continue
10720         }
10730       }
10740       freads( line$, nfp )
10750       if line$=chr$(12) then break
10760       locate dataEntry( j ), j
10770       print left$( line$, 95-dataEntry( j ))
10780       Data( j ) = line$
10790     next
10800     writeCard( newCard() )
10810     usingCards = usingCards + 1
10820   next
10830   readCard( cards( 0 ))
10840 endfunc
10850 /*
10860 /*  1行入力
10870 /*
10880 func str getLine( msg;str )
10890   str tmp, ch
10900   locate 0,31
10910   print msg;chr$(5);
10920   ch = ""
10930   while ch<>chr$(13)
10940     ch = inkey$
10950     if ch = chr$(8) then {
10960       tmp = left$( tmp, strlen( tmp )-1 )
10970       locate pos-1, csrlin
10980       print " ";
10990       locate pos-1, csrlin
11000     } else if ch >= " " then {
11010       print ch;
11020       tmp = tmp+ch
11030     }
11040   endwhile
11050   return( tmp )
11060 endfunc
```

カード型データベースCBASE.Xリファレンスマニュアル

CBASE.Xの動作環境

Human68kのコマンドモードまたは，VS2.X（付録ディスクに収録）によって拡張されたビジュアルシェル上で動作します。

ただし，付録ディスクには容量の都合で日本語フロントエンドプロセッサASK68K.SYSは収録されていません。解凍したディスク1を利用して日本語入力を行う場合には，以下の手順に従ってASK68K.SYSを組み込んでください。

1)　お手持ちのシステムディスクからASK68K.SYSをコピーする。

ASK68K.SYSは，X68000付属のシステムディスクのSYSというディレクトリに入っています。ドライブ0からシステムディスクを起動し，コマンドモードに入ってください。ドライブ1にディスク1を入れて，

A>COPY A:¥SYS¥ASK68K.SYS B:¥SYS

と入力しリターンキーを押します。

2)　CONFIG.SYSにASK68K.SYSを登録する。

まず，エディタED.Xを起動します。

A>ED B:¥CONFIG.SYS

CONFIG.SYSの内容が表示されたら適当な位置に次の1行を追加します。

DEVICE = ASK68K.SYS B:¥X68K_M.DIC B:¥X68K_S.DIC

これは，辞書をBドライブで使用する場合です。

ESCキーに続いてEキーを押すとセーブしてエディタを終了します。

以上で準備は完了しました。ディスク1をドライブ0，辞書ディスクをドライブ1に入れてリセットボタンを押してください。

＊

【起動方法】

cbase 〔データベースファイル名〕〔0〕

・データベースファイル名は拡張子を付けずに指定する。データベースファイル名を省略した場合は，起動後ファイル名の入力となる。

・オプションとして0を指定すると，VS2.Xから起動した場合と同様に12ドットフォントで表示される。ただし，終了後カーソルは表示されない。

【CBASE.Xの能力】

CBASE.Xは最大で2000データ，250枚のカードを扱うことができる。この講座で作成したデータベースよりカード数を150枚増やしてあり，設計情報ファイル（*.fmt）を除いてファイルに互換性はない。ただし，CBASE.XではX-BASIC版データベースで作成したファイルも扱うことができる。この場合カードの最大数は100枚である。

【データベースの設計】

起動時あるいは起動後に指定したデータベースファイルが存在しない場合，新規データベースと見なしてデータベースの設計画面になる。設計画面で使える編集キーは以下のとおり。

←	カーソル左
→	カーソル右
↑	カーソル上
↓	カーソル下
BS	カーソル左の文字を消去
DEL	カーソル位置の文字を消去
ROLL UP	カーソル行を消去し，以下の行を1行上へ
ROLL DOWN	カーソル位置に1行挿入し，以下の行を1行下へ

データ入力位置は'#'で指定する。データ入力位置は1行1カ所に限られる。たとえば，

名称　　#

という行を作成すると，実行時には，

名称　　■

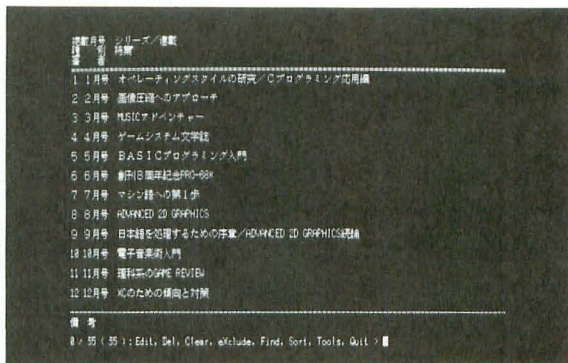
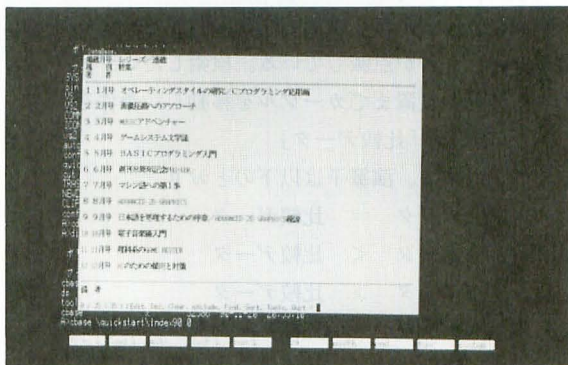
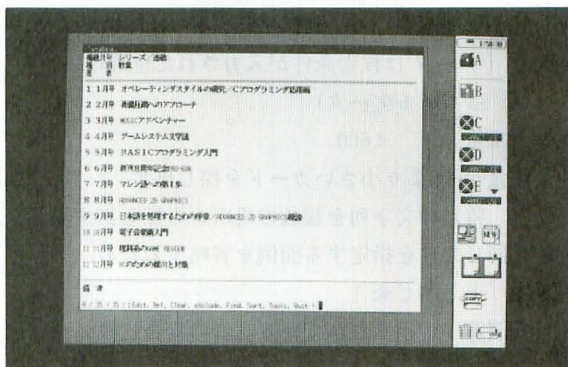
のようにカーソルが'#'位置に表示されデータ入力できるようになる。

データベースの設計は，ESCキーを押すと終了する。データベース用ファイルの作成が開始され，しばらく待つとデータ閲覧画面になる。

【データの閲覧】

データ閲覧画面では，画面最下行に，

0 / 30 (40) Edit, ……

という形式のメニューが表示される。カッコ内の数字は現在使用中のカード数，その左の数字は使用中のカードから閲覧用にピックアップされているカード数，最も左の数字はピックアップされたカード中の何番目のカードを表示しているかを示している。カードの閲覧はROLL UP，ROLL DOWNキーで行う。ピックアップされているカード以外は閲覧できない。

標準的な画面モード。ディスク1のサンプルデータindex90.dat場合には，コマンドモードから，
cbase index90
で起動する

12ドットフォントの文字を使用するモード。
cbase index90 0
で起動する

ビジュアルシェル上で動くモード。解凍したディスク1では，
index90.～
のアイコンをダブルクリックすると起動する

メニューは最初の英大文字をキーボードから入力することによって選択される。以下にその機能を挙げる。

●**カードの修正（E：Edit）**

表示されているカードへのデータ入力，データ修正を行う。ESCキーを押すと入力･修正は終了する。Editメニューではカーソル上下左右，リターン，BS，DELキーが編集用に使用できる。入力された文字は常にカーソル位置へ挿入される。

●**カードの削除（D：Del）**

表示されているカードを削除する。削除の確認を求め，'y'キーが押された場合のみ削除を行う。

●**ピックアップカードの破棄（C：Clear）**

ピックアップカードを無効とし，再び使用中の全カードをピックアップカードとする。

●**カードの除外（X：eXclude）**

表示中のカードをピックアップカードから外す。カードの削除ではない。

●カードの検索（F：Find）

条件を指定してカードの検索を行う。選択すると，

Normal，Additional >

というメニューが表示される。これは検索モードの指定で，Normal検索はピックアップされたカードの中から条件に合致するものを探し出しピックアップカードとする。Additional検索は，ピックアップされていないカードの中から条件に合致するものを探し出しピックアップカードに加える。

検索モードを指定すると条件入力画面となる。条件入力画面はカードの修正画面と同様の画面で，データ入力位置でカーソルが点滅している。検索したいデータが入力されている位置までカーソルを移動し，条件を，

「演算子」「比較データ」

の形で入力する。演算子は以下のとおり。

=：　データ　=　比較データ
<：　データ　<　比較データ
>：　データ　>　比較データ
!：　データ　≠　比較データ
(：　データ　⊆　比較データ
)：　データ　⊇　比較データ

（‘データ’は検索条件が入力された位置に入っているデータ）

例）　重量　　<600

重量が600より小さいカードを探し出す

また，特定の文字列を複数の場所から探し出す場合に，何度も同じ条件を指定する面倒を省略するため，

^：　上と同じ条件

という演算子が用意されている。この場合比較データは必要ない。比較はすべて文字列比較で行われるので注意されたい。

検索条件の入力はESCキーで終了する。条件入力が終了すると，検索方法が画面最下行に表示される。

And，Or >

Andは入力したすべての条件を満たすカードを検索する。Orは入力したすべてのいずれかを満たしているカードを検索する。

●カードの並べ替え（S：Sort）

ピックアップされているカードの並べ替えを行う。昇順・降順の並べ替えが可能で，検索条件を入力したときと同様に，並べ替えのキーとなるデータが入力されている場所に「<：昇順」「>：降順」を書き込みESCキーを押すと並べ替えられる。複数キーによる並べ替えはできない。最初に指定された場所だけが有効である。

●ツール（T：Tools）

テキスト入出力，カードの再設計を行うツール選択メニューを起動する。

●終了（Q：Quit）

カード型データベースを終了する。

【ツールの使い方】

Toolsメニューによってツールが起動されると，次のメニューが表示される。

OutputText，InputText，EditFormat >

ESCキーで終了。以下にその使い方を説明する。

●テキスト出力（O：OutputText）

データベースファイルを，ワープロやエディタで利用可能なテキストファイルとして出力する。ファイル名を入力すると，各カードのそれぞれのデータを改行で区切り，カードを改頁で区切ったテキストファイルが生成される。出力されるのはピックアップされているカードだけである。

●テキスト入力（I：InputText）

データを改行で，カードを改頁で区切ったテキストファイルを読み込み，データベースファイルに追加する。ファイル名を入力すると，データの読み込みが行われる。読み込まれたデータはピックアップカードには自動追加されないので，Clearメニューなどを利用してピックアップする必要がある。

●カードの再設計（E：EditFormat）

EditFormatを実行するとカードの設計画面が再表示され，再設計用メニューが画面最下行に現れる。

Move，Edit，Update >

再設計はESCキーで終了する。

Moveは行の移動を行う。最初に，

移動するレコード指定

と表示されるので，反転表示された行をカーソル上下で移動し，動かしたい行でリターンキーを押す。次に，

移動先指定

と表示されるので，同様にして移動先を指示する。指定された行は削除され，移動先に挿入される。ESCキーで再設計メニューに復帰する。

Editは行の修正を行う。カーソル左右，BS，DEL キーが編集用に使用できる。修正はリターンあるいはESCキーで終了し，再設計メニューに復帰する。データ入力位置を示す‘#’を削除した場合は，Updateするまでその行に‘#’を書き込むことはできない。

Updateは設計の更新を行う。行の移動，‘#’の削除に対応して入力されているデータの移動・削除も同時に行われる。誤って設計情報から‘#’を削除してしまった場合は，Updateせずに ESCキーでツールメニューに復帰すれば元の設計情報が復帰する。

【付属データ】

サンプルデータファイルとして，年末インデックスの一部がディスクに付属している。収録されているのは特集と連載だけであるが，これを参考に自分のデータベースを構築してみていただきたい。カードの設計方法は，

Tools-EditFormat

で表示されるインデックスの設計を参照されたい。

シミュレーションへの道も絵描きから

前回X1 BASICで記述していた高速道路の料金所シミュレーションプログラムをX68000のX-BASICで記述してみましょう。まだ耳に馴染んでいない言葉が出てくると思いますので，事前に前回の記事を読み返しておくことをおすすめします。

Kamon Masato 華門 真人

Experience

昨年の春（というとまぎらわしいが1990年の春のことである），僕はアメリカに住んでいる両親のもとを訪ねた。そこでひとつの面白い事実を発見した。

発見は例によってクルマに乗っているときのことである。クルマとはいってもアメリカでのことであるから，もちろんアルシオーネではない。そのときはビュイックのパークアベニューという，いわゆるフルサイズのアメ車，しかも10年くらい前の古き良き時代の，に乗っていた。親父の会社のクルマを借りていたのだ。

いい忘れていたが，僕の両親はテキサス州のヒューストンに住んでいる。ヒューストンというと夏に開催されたサミットのことを思いおこされる博識な方もいらっしゃるだろう。確かにヒューストンはサミットシティとして一時的には有名になったが，実際のところはあまり知られていないように思う。

実はヒューストンはブッシュ大統領の地元であり，全米第4の大都市でもある。ほら，驚いたでしょう。かくいう僕自身も実際に行ってみるまでは，テキサスというぐらいだから砂漠の中にぽつんとある街なのかと思っていた。

実際のところは緑が豊かで広々とした都会であった。いやぁいいところですよ，本当に。ただこれはアメリカのほとんどの都市にいえることだけれども，あまりにも広々としているうえに公共交通機関もほとんどないから(バスぐらいだ)，どこに行くにも，なにをするにもクルマがないと生活できない。

というわけでビュイックにはお世話になった。これではひとりに1台クルマがないと生活していけないなと実感した次第。しかし逆にクルマがそれこそ「ゲタ」代わりになっているだけあって，道路網は素晴らしく発達している。一般道はたいてい片側3，4車線と広々しているし，高速道路も網の目のように張りめぐらされている。日本でいうと首都高速のようなものだけれど，どこでも最低片側3車線はあるし，しかも大部分は「タダ」。

まったく羨ましくなってしまうぐらいの充実ぶりである，特に僕のようなクルマ好きには。でもそんな広々とした道路も，ラッシュアワーには大渋滞している。さすがはクルマ大国というべきか。

本題に戻ろう。僕が興味深い発見をしたのは，ガルベストンというヒューストンから50マイル（約80km）ぐらい離れた港町に遊びに行った帰りのことである。行きは下でのんびりと行ったのだが，帰りは疲れていたこともあってI-45（インターステート45号線）で素直に帰ろうということになった。

I-45は快適そのものであった。なんといってもアメリカ，道路は広いし，こちらはまるで戦艦のようなアメ車である，乗り心地は抜群。と，前方で工事のようである。ああ，やっぱり日本と同じでアメリカでも道路工事ぐらいあるのだ。しかし不思議なことに，クルマの流れが悪くなるようなことはない。

すぐにその理由がわかった。工事区間に近づくにつれ徐々に車線が狭くなっていくのである。もともとアメリカの車線はゆったりしているのだが，それがあっという間にクルマぎりぎり1台分にまでなってしまった。

しかしその反面，車線は減少しない。だから合流などによって渋滞することもないということだったのだ。

ここでようやく問題は核心にせまる。日本ではよく工事で渋滞してしまうことが多い。いつもはすいている道路がその日に限って大渋滞，なぜなんだろうと思っていたら実は工事だった，という経験をした人も多いと思う。

Observation

同じ工事でもアメリカでは渋滞にはならないのに，日本では渋滞する。その鍵は道路のキャパシティ（容量，嚙みくだいていうならば処理能力）にある。アメリカでは車線1つひとつの幅は狭くなっても車線の数自体は減らない，だから道路のキャパシティは工事中でも減ることはない。

ひるがえって日本では，工事というとすぐ車線規制をして，車線数を減らしてしまう。こうなると工事によって道路のキャパシティが減ってしまうことになる。その結果どんどんやってくるクルマを処理しきれなくなって渋滞が起こる，ということなのだ。

もっと身近な例でいえば，スーパーマーケットのレジなんかを考えてみてもいい。レジがあまりあいていないと，数少ないレジにお客が集中するものだから，レジに長蛇の列なんてことも珍しくない。

もっともそんなことじゃあ，お客様商売としては失格だよね。そこで店のほうも，レジの数を増やすなどして対処しようとするわけ。すると列がたちまち短くなったりする。

結局これはレジのキャパシティの問題といえる。レジの数が少ないと，当然同時に処理できる客の数が少ないということだから，いい換えればキャパシティが小さい，ということになる。

ところがレジの数を増やせば，同時に多くの客を処理することができる。それがすなわちキャパシティが大きくなる，ということなわけだ。

もっとも，混むかどうかというのはキャパシティだけによるわけではない。処理対象がどれだけあるか，ということにも大きく左右される。

これは少し考えてみればわかることだろう。たとえば1時間に2,3人しかお客のこな

いスーパーであれば，どう考えてもレジは1台あれば十分だろう（スーパーというよりも個人商店というべきか）。

ところが1時間に何百人，何千人単位でお客がくる大きなスーパーでは（ちなみに最近はこういった規模の大きいスーパーをハイパーマーケットと呼ぶらしい，う～んハイパー），相当多くのレジを設置しないと，処理が追いつきそうもないというのは自明だろう。

さらに付け加えるならば，キャパシティは簡単な計算によっても考えてみることができる。前回の料金所モデルで考えてみよう。まず改良前のモデルの1分間を想定する。平均すると，この1分間に20台のクルマが流入してくる。このうち18台が右ハンドルで，2台が左ハンドルである。2台の左ハンドルは発券機3で14秒あれば処理できる。

ところが右ハンドルはというと，発券機2台をもってしても63秒（＝18×7÷2）かかってしまう。すなわち1分間に流入してくるクルマを，1分間では処理しきれないのである。つまりパンク状態。大渋滞を起こすのも無理はない。一方改良後は，同じ18台の右ハンドルでも，発券機3も利用することにより処理しきれてしまう。

とまあ，僕の観察したちょっとした事実から渋滞発生の理論にまで話を拡大してみた。おそらく日本では安全第一ということで車線規制をするのであろうが，こんなことからも日本人とアメリカ人の考え方の違いが垣間見えて面白いものである。こう考えてみるとなかなか渋滞っていうのも身近なものに思えるかも（あまりありがたくない身近なものではある）。

ところで，アメリカでの話にはちゃんとしたオチがある。うん，これならば渋滞にはならないな，と感心したのはいいのだが，忘れてはいけない，僕がそのとき乗っていたのはアルシオーネではない，ボクシングでいうなら超ヘビー級のビュイックだ。

皆さんはいわゆるフルサイズのアメ車に乗ったことがあるだろうか。これがとにかく大きいのである。残念ながら正確な大きさは忘れてしまったのだが，そのボリューム感はベンツ560（いわゆる大ベン）をも凌ぐものがある。

もっとも僕でさえはじめは尻込みしてしまった巨大なビュイックであるが，いったん乗りだしてしまえば特に困ることもなかった。なぜならアメリカでは道路が十分に広いからである。

しかしそれも道路が広いときの話。ひとたび工事となると，あれほど広かった車線も急激に狭くなる。巨大なビュイックはほとんど車線をはみださんとする。ドライバーとしてはとにかく車線をはみださないようにするだけで精一杯。そこを左右から他のクルマが追い越しをかけていく。

はっきりいってあのときは冷えた。ハンドルを握ったままほとんど硬直。汗はどっと出るわ，目は血ばしるわ，それはもう僕

リスト1

```
1000 /* Simulation model 2 ver.1.01
1010 /*
1020 /*          for X-BASIC
1030 /*
1040 /*              1990.10 (c) Cammon
2000 /*
2010 /* initialize : set constant
2020 width 96
2030 int t,nxact
2040 char pgen1=3,pgen2=2
2050 char ptran=1
2060 dim char p2(3)={ 0,7,7,7 }
2070 dim char p3(3)={ 0,2,2,2 }
2080 int oxact
2090 int dxactt,i
2100 char xact,p1
2110 dim int u(3),q(3)
2120 dim int quwtt(3),qfreet(3),qfree(3)
2130 dim int xacttt(3),dxact(3)
2140 dim int nque(3),nqze(3)
2150 dim int nfce(3),maxq(3)
2155 str in     /* for c
2160 input "Number of Xact : ",oxact
2170 cls
3000 /*
3010 /* main
3020 while 1
3030   locate 0,0: print using "  t   : #####";t
3040   if t=nxact then xact=generate()
3050   for i=1 to 3
3060     queue()
3070     if dxactt=oxact then break
3080   next
3090   locate 0,1: print using "Xact : #####";dxactt
3100   if dxactt=oxact then break
3110   t=t+1
3120 endwhile
3130 result()
3135 in=inkey$  /* for c
3140 end
10000 /*
10010 /* generate Xact
10020 func char generate()
10030   nxact = t + pgen1-pgen2+int(rnd()*(2*pgen2+1))
10040   if int(rnd()*10) < ptran then {
10050     p1=3 } else { p1=gensub() }
10060   return(1)
10070 endfunc
10080 /*
10090 /* generate sub
10100 func char gensub()
10110   char p
10120   p= int(rnd()*2)+1
10130   if u(1)=0 and u(2)=0 then return(p)
10140   if u(1)=0 and u(2) then return(1)
10150   if u(1) and u(2)=0 then return(2)
10160   if q(1)=q(2) then return(p)
10170   if q(1)>q(2) then return(2)
10180   return(1)
10190 endfunc
20000 /*
20010 /* queue
20020 func queue()
20030   quwtt(i)=quwtt(i)+q(i)
20040   if xact and p1=i then xact=mkque(): return()
20050   if u(i)=0 or t<qfreet(i) then return()
20060   xacttt(i)=xacttt(i)+qfree(i)
20070   if q(i) then q(i)=q(i)-1: mkqfr() else u(i)=0
20080   dxact(i)=dxact(i)+1
20090   dxactt=dxactt+1
20100   return()
20110 endfunc
20120 /*
20130 func char mkque()
20140   nque(i)=nque(i)+1
20150   if u(i) then {
20160     q(i)=q(i)+1
20170     if q(i)>maxq(i) then maxq(i)=q(i)
20180     return(0)}
20190   u(i)=1: nqze(i)=nqze(i)+1
20200   mkqfr()
20210   return(0)
20220 endfunc
20230 /*
20240 func mkqfr()
20250   qfree(i)=p2(i)-p3(i)+int(rnd()*(2*p3(i)+1))
20260   qfreet(i)=qfree(i)+t
20270   nfce(i)=nfce(i)+1
20280   return()
20290 endfunc
25000 /*
25010 /* print result
25020 func result()
25030   print: print "Result": print
25040   print "Simulation time (t) :";t;"s": print
25050   print " -- Facility --"
25060   print "   name        average        number        average"
25070   print "   or no.  utilization        entries   time / Xact"
25080   for i=1 to 3
25090     print using "     #";i;
25100     print using "          ##.####";1#*xacttt(i)/t;
25110     print using "           #####";nfce(i);
25120     print using "        ###.###";1#*xacttt(i)/dxact(i);
25130     if u(i) then print "              in use";
25140     print
25150   next
25160   print: print: print " --  Queue  --"
25170   print "  no.     max     average     total     zero    perc
ent  average  current"
25180   print "             contents   contents  entries  entries  zer
o   time /Xac  t  contents"
25190   for i=1 to 3
25200     print using "     #";i;
25210     print using "      ###";maxq(i);
25220     print using "       ###.###";1#*quwtt(i)/t;
25230     print using "     #####";nque(i);
25240     print using "     #####";nqze(i);
25250     print using "    ##.###";1#*nqze(i)/nque(i);
25260     print using "    ###.###";1#*quwtt(i)/nque(i);
25270     print using "          ###";q(i)
25280   next
25290   print
25300 endfunc
```

にしてみれば大変な騒ぎだったのである。が，あとで同乗者に聞いてみたところ，なにも気がついていなかったようである。羨ましい。

このような「恐怖の」経験をしたのは僕だけではない。どこかのクルマ雑誌で，ドイツ現地取材という記事があったのだが，ドイツでもやはり工事の際は車線幅を狭くして車線数を維持するらしい。その記事の筆者によると「とても怖かった」ということだ。

Groundwork

さて，前置きはこれぐらいにして本題に入る。前回はコンピュータシミュレーションの有効性について見てきたわけだが，約束どおり，今回はどのようにしてプログラムにするか，ということを考えていこうと思う。

題材としてはリスト1を用いることにする。これは前回のモデルをX-BASIC上で実現させるためのもの。先月のX1用と基本的に一緒と考えてもらって差し支えない。

念のためにモデルの解説をしておこう。モデルの舞台となるのはある仮想の料金所である。この料金所での通行券の発券をシミュレートする。

シミュレーションの前提は以下のとおりである。

・発券用の車線は3車線あり，そのうち2つは右ハンドル専用（第1,2レーン），残りのひとつが左ハンドル専用（第3レーン）である。

・発券にはどの発券機でも7±2秒かかり，クルマは3±2秒間隔でやってくるとする。なお，クルマ全体に占める左ハンドルの割合は10%であるとする。

・料金所に入ってくるクルマは左ハンドルならば無条件に第3レーンに進む。右ハンドルは，第1レーンか第2レーンのうち，すいているほうに進む。もし同じように混んでいるときは等確率で第1レーンか第2レーンのどちらかに進む。

あるシミュレーションモデルをプログラムとして実現する際，まずなにから始めればいいのだろう。当たり前のことかもしれないが，まずモデルをしっかり把握することから始めなければならない。

まずはモデルを図にして要点を把握することだ。まさしくシミュレーションの道もお絵描きから始まるのだ。図1（前回の図3と同じもの）がこれにあたる。トランザクション（処理対象）の動き，あるいはファシリティ（処理する施設）との関係などをしっかり把握しよう。

把握できたところでフローチャートを作成する。そしてそのフローチャートをもとにプログラムにする。そう，普通のプログラムを作成するのと基本的には変わらないのだ。

もっとも，フローチャートといっても少し独特なものを利用することにする。それでは図2を見てほしい。これはトランザクション（このモデルの場合，クルマ）を中心にしてモデルの流れを言葉で書いてみたものである。

まず初めに料金所にクルマが流れ込んでくる。ただ，クルマといっても右ハンドルと左ハンドルの2種類ある。そこで発券機の選択を行う必要がでてくる。

すなわち，左ハンドルならば無条件に発券機3を選択する。一方，右ハンドルならば，発券機1か2のどちらかを選択することになる。そこで右ハンドルの場合のみ，発券機1,2の選択という項目が加わることになる。

さて，クルマごとに発券機が決まった。ここでようやく本来の目的である「発券」という作業ができることになる。ただ，ひと口に発券とはいっても，場合によっては待ち行列のことなども頭に入れておかなければならない。

かくのごとき手間を経てようやく，クルマを料金所から開放することができる。もちろんその前に，結果を得るための計算などもそつなくこなしておかねばならない。

図1　料金所の様子

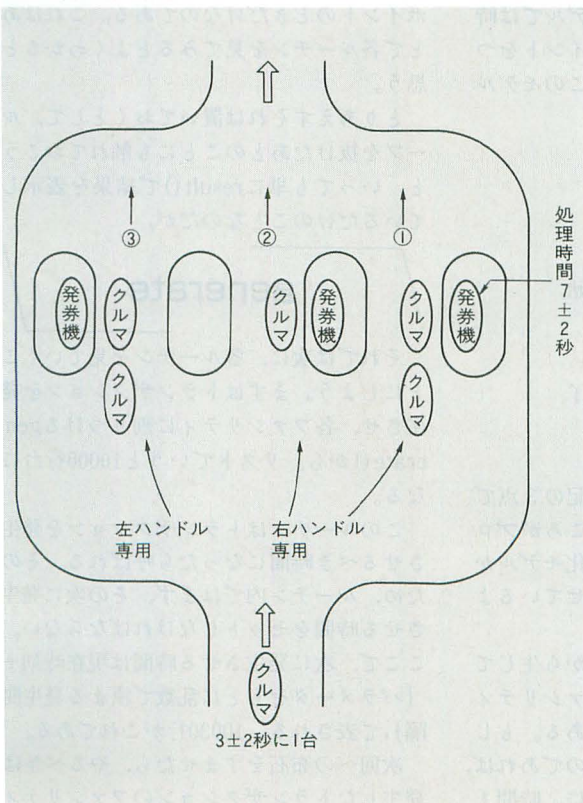
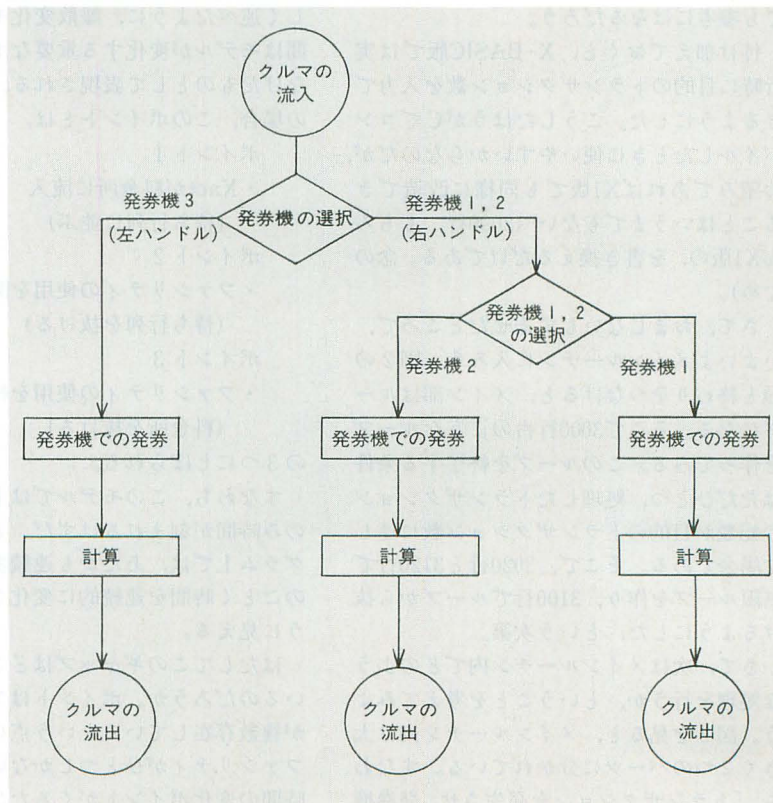



図2　料金所モデルのフローチャート

以上が図2のフローチャートの概要というわけだ。こうして言葉で表現してみるとモデルを的確に，しかも容易に把握することができる。何事も最初はシンプルにという法則は，ここでも生きているのだ。このシンプル極まりないチャートが，実際にプログラムを作るときの土台になる。

以下では，図2とリスト1を対比させながら説明していくことにしよう。多少細部に違いはあるが，先月のX1版でも基本は同じである。

main {

およそプログラムであるかぎり，まずは初期設定というものが必要になる。まあよくいわれるところの「おまじない」みたいなもの。リスト1では2000行台がこれに該当する（以下，行番号は特に指定がない限りリスト1を指す）。

やっていることは変数の宣言と初期値のセットぐらいのもの。参考までに変数名のだいたいの基準を述べておこう。まず変数名にpが含まれている場合はパラメータを表している。同様にしてxactはトランザクションに関する変数を，qは待ち行列（キュー）に関する変数を表している。

残念ながら筆者の「行きあたりばったりな」性格（？）を反映してか，あまりわかりやすい変数名にはなっていないが，これでも参考にはなるだろう。

付け加えておくと，X-BASIC版では実行時に目的のトランザクション数を入力できるようにした。こうしたほうがCでコンパイルしたときに使いやすいからなのだが，お望みであればX1版でも同様に改造できることはいうまでもない（2090行，もちろんX1版の，を書き換えるだけである。念のため）。

さて，おまじないもすませたところで，いよいよメインルーチンに入ろう。図2の頭と終わりをつなげると，メイン部はループになる。そこで3000行台のようなループを作ってみる。このループを終了する条件はただひとつ，処理したトランザクションの総数が目的のトランザクション数に達した場合である。そこで，3020行と3120行で無限ループを作り，3100行でループから抜けるようにした，という次第。

さて，次はメインルーチン内でどのような処理を行うか，ということを考えてみよう。図2を見ると，メインルーチン内は大きく2つのパーツに分かれている。すなわち，「トランザクションを発生させ，発券機に割りあてる」パーツと，「待ち行列などの，ファシリティでのトランザクションの処理と計算を行う」パーツである。

これら2つのパーツをそれぞれサブルーチンとして処理してしまうと，メインルーチンは非常に単純になる。もうおわかりだと思うが，この2つのパーツは前者がgenerate()，後者はqueue()となっている。

以上を頭に入れて，メインルーチンの動きを追ってみよう。まず，時刻を調べ，新しいトランザクションを発生させるべきかをチェックする。もし新しいトランザクションを発生させるのであれば，generate()を呼ぶ。

次に，各ファシリティ（1～3）に対しqueue()をコールする。このqueue()で待ち行列およびファシリティのコントロールを行うわけだ。この時点で処理されたトランザクションの数が目標値に達すると，ループから抜け出す。

しかしたいていの場合は時間を経過させたあとに，ループの頭に戻ることになる。ここで疑問を感じた人。あなたは鋭い。というのもこのモデルは離散変化モデルのはずだからである。このループでは時間を中心にモデルが動いていっているように見えないだろうか。もしそうならば，これは連続変化モデルになってしまう。

確かにこのモデルは離散変化モデルだし，図2もそのように書かれている。前回で詳しく述べたように，離散変化モデルでは時間はモデルが変化する重要なポイントをつなげたものとして表現される。このモデルの場合，このポイントとは，

　ポイント1
　・Xactが料金所に流入
　　(待ち行列に並ぶ)
　ポイント2
　・ファシリティの使用を開始
　　(待ち行列を抜ける)
　ポイント3
　・ファシリティの使用を終了
　　(料金所を抜ける)

の3つにしぼられる。

すなわち，このモデルでは上記の3点でのみ時間が刻まれるはずだ。ところがプログラム上では，あたかも連続変化モデルかのごとく時間を連続的に変化させているように見える。

はたしてこのギャップはどこから生じているのだろうか。ポイントはファシリティが複数存在しているという点にある。もしファシリティがひとつしかないのであれば，時間の変化ポイントがくるたびに，時間tに増分を加えるかたちで時間を変化させればよい。これが「本来の」離散変化モデルであるし，問題はまったくない。

ところがファシリティが今回のように複数になると，話は異なってくる。すなわち各ファシリティごとに現在時刻が異なってきてしまう，という問題が生じる。つまりこういうことだ。ファシリティ1,2,3それぞれでそれぞれのポイントに従って独自に時間を変化させていく。そうすると当然，ある時点で各ファシリティの様子を見てみると，各ファシリティでそのときの時刻が違ってくる。

これは実に不都合な状態である。たとえばある時刻に新しいトランザクションが発生したとする。そのとき，どのファシリティの待ち行列に並ぶかを決めようにも，各ファシリティで時間が違うから，その時刻の各ファシリティの様子を知ることができない。その結果トランザクションがどのファシリティに属するかも決定できない，という事態になってしまう。

そこで今回のようなプログラムでは，便宜的に時間を中心にループをまわすという方法をとるというわけ。しかし，これでも離散変化モデルである以上，連続変化モデルとは決定的に異なる点がある。

すなわち，ループ内でこそ時間tは1刻みで増加していくが，ファシリティの観点に立ってみると，時間が変化するのは変化ポイントのときだけなのである。これはあとで各ルーチンを見てみるとよくわかると思う。

とりあえずそれは置いておくとして，ループを抜けたあとのことにも触れておこう。と，いっても単にresult()で結果を表示しているだけのことなのだが。

generate

それでは次に，各ルーチンを見ていくことにしよう。まずはトランザクションを発生させ，各ファシリティに割りつけるgenerate()から。リストでいうと10000行台になる。

このルーチンはトランザクションを発生させるべき時間になったら呼ばれる。そのため，ルーチン内ではまず，その次に発生させる時間をセットしなければならない。ここで，次に発生させる時間は現在時刻＋{パラメータをもとに乱数で決まる発生間隔}で表される。10030行がこれである。

次回への布石をすませたら，やるべきは発生したトランザクションのファシリティ

への割りつけである。そのためには乱数によって右/左ハンドルを決定する。もし左ハンドルであれば，ファシリティは3に決定（p1＝3）である。

ところが，右ハンドルの場合は，ファシリティ1/2の混み具合を見たうえでどちらかを選択しなければならない。これを一手に引き受けているのがgensub()というわけだ。gensub自体は単に条件を羅列しているだけだから，自分でフォローしてみてほしい。

queue

お次にひかえますは，待ち行列やファシリティの状態をコントロールする関数，queue()である。queueはファシリティごとに実行されるわけだが，まずそのファシリティで新たにトランザクションが発生したかどうかをチェックする。

トランザクションが発生していたらmkque()に飛ぶ。mkqueでは現在ファシリティが使用中かどうかを調べなければならない。使用中であれば話は簡単。待ち行列の最後尾に加えてやればいいのだ。

使用中でなかった場合。この場合，処理はmkqfr()にまかされるが，やっていることはファシリティでの処理時間が何秒になるかを決定するだけの話である。この処理時間もパラメータをもとに乱数で決定される。もちろんファシリティを使用中に変更してトランザクションの発生処理は終わる。

逆に，トランザクションが発生しておらず，また，ファシリティが使用中でもない場合はなにもせずに帰ればよい。ファシリティが使用中でも，まだ使用が終わらない場合も同様である。話が少し複雑になるのは，ファシリティの使用がちょうど終わった場合である。

この場合もそのファシリティに待ち行列が存在しているかどうかで，話が変わってくる。まずは待ち行列が存在しない，すなわち後ろに誰も待っていない場合を考えてみよう。こちらは話が簡単で，ファシリティを空きにして（u(i)＝0）リターンする。

一方，待ち行列が存在している場合は，次の番へ順番を回してやらなければならない。そして新たにファシリティを使用することになったトランザクションに対し，mkqfrで使用時間を設定する。さらに最後に待ち行列の長さを1だけ減らしてリターンする，というわけである。

付け加えるとこのqueueルーチンでは，このような処理と同時に，結果を得るための計算も行っている（というか，こちらが本題ともいえるのだけれど)。これに関しては表1を参考に自分で考えてみるといいだろう。

さて，こうやって見てきて，サブルーチン内ではモデルの重要変化ポイントで時間が刻まれているということ，おわかりいただけました？

Result

こうして丁寧に見ていくと，案外シミュレーションプログラミングというものが簡単だということがわかるだろう。意外かな。一見複雑そうに見えるシステムもうまくモデル化してやれば簡単にシミュレートすることができるのだ。

ところで，前回はこのモデルにさらに改良を加えてみた。すなわち，第3レーンを右ハンドルでも左ハンドルでも利用できるように変更した。X-BASIC上でも同じように改良してみることにしよう。リスト2をリスト1につぎはぎすれば出来上がりだ。行番号でいうと，10100行から10190行を書き換え，30000行以降を加えればいい。これはルーチンでいえばgensubを書き換えていることになる。すなわち右ハンドル時のファシリティ選択ルーチンだけを書き換えているというわけ。

それでは改良前と改良後，それぞれを実行してみることにしよう。今回はトランザクションの目標値を10000として実行してみた。これもX-BASIC版ではコンバータを通すことによってCでコンパイルすることができるからである。

前回のX1版を実行してみて実行速度に不満を持たれた方も多いであろう。確かにあれは遅い。とてもじゃないが目標値1000以下でしか実行する気になれない。あれもいろいろとキタナイことをすれば多少は速くなるのだろうが，多少の速さよりプログラムの素直さを重視させてもらった。

そのストレスを発散すべく，今回のプログラムをコンパイルして実行したものは速い。これならば気長に10000回待ってもいいか

表1　統計計算式表

▶average utilization

$$=\frac{(\text{ファシリティ使用時間総計})}{(\text{全シミュレーション時間})}$$

＝xacttt (i)/t

▶average time/Xact (facility)

$$=\frac{(\text{ファシリティ使用時間総数})}{(\text{ファシリティで処理したXact数})}$$

＝xacttt (i)/dxact (i)

▶average contents

$$=\frac{\Sigma\{(\text{待ち行列に入っていた時間})\times(\text{トランザクション個数})\}}{(\text{全シミュレーション時間})}$$

＝quwtt (i)/t

▶perecent zero

$$=\frac{(\text{zero entries})}{(\text{total entries})}$$

＝nqze (i)/nque (i)

▶average time/Xact (queue)

$$=\frac{\Sigma\{(\text{待ち行列に入っていた時間})\times(\text{トランザクション個数})\}}{(\text{total entries})}$$

＝quwtt (i)/nque (i)

リスト2

```
10100 func char gensub()
10110   char p
10120   p= min(u(1),u(2),u(3))
10130   if p>3 then return(p-3)
10140   if p then return(grnd(p))
10150   if u(1)=0 then return(grnd(0))
10160   p= min(q(1),q(2),q(3))
10170   if p>3 then return(p-3)
10180   return(grnd(p))
10190 endfunc

30000 /*
30010 /* minimum
30020 func char min(x;int,y;int,z;int)
30030   if x<y and x<z then return(4)
30040   if x<y and x>z then return(6)
30050   if x<y and x=z then return(2)
30060   if y<z and y<x then return(5)
30070   if y<z and y>x then return(4)
30080   if y<z and y=x then return(3)
30090   if z<x and z<y then return(6)
30100   if z<x and z>y then return(5)
30110   if z<x and z=y then return(1)
30120   return(0)
30130 endfunc
35000 /*
35010 /* generate rnd
35020 func char grnd(x;char)
35030   char r
35040   switch x
35050     case 0 :r=int(rnd()*3)+1     : break
35060     case 1 :r=int(rnd()*2)+2     : break
35070     case 2 :r=(int(rnd()*2))*2+1: break
35080     case 3 :r=int(rnd()*2)+1     : break
35090   endswitch
35100   return(r)
35110 endfunc
```

なという気にさせてくれる。さすがはCコンパイラ，ありがたや。

まあそれはともかくとして，実行結果を見てみよう。表2が改良前，表3が改良後である。目標値を大きくしても傾向は変わらないことがわかるだろう。

and……

以上でどのようにしてプログラムを作成するか，ということを見てきた。割合と難しいな，と感じた方もいらっしゃるだろうし，なんだ簡単じゃない，と思われた方も多いだろう。もちろんそれは個人差なのだけれど，仮に簡単だと思った人でも，モデルを大幅に改造しようとするとそれなりの苦労をすることになるだろう。

え，でも今回の改良は簡単だったじゃない，だって。それはこのプログラムができるだけフレキシブルに書いてあったからの話。とはいえフレキシブルにも限界がある。ある程度以上の変更をしようとすると，なかなか大変な事態になるだろう。

たとえばの話，この料金所に救急車のような緊急車両がやってきたらどうするか。当然待っているクルマを追い越して緊急車両を通過させなければならない。すなわち待ち行列に対する「割り込み」が生ずることになる。ここまでくると今回のプログラムで対処することは，不可能とはいわないまでもかなり難しいだろう。

それではどうするか。ひとつの一般的な道として，「それじゃあシミュレーションを実行するための専用言語を作っちゃえ」という考え方がある。このような考え方をした人は多いらしく，有名な言語としてGPSS，SIMSCRIPT，SOL，そしてSIMULAなどがある。

これらの言語は実は歴史が長く，1961年にはもうGPSSそしてSIMSCRIPTが開発されている。このなかでもいちばんメジャーなのがGPSS (General Purpose Simulation System)である。

ちなみに今回のモデル (改良前) をGPSSで書いてみるとリスト3のようになる。ひと目見ればわかるとおり，たったの20行ほどである。BASICだと最低でも100行はかかるのに，である。もっとも，それが専用言語の専用言語たるところなのだろうが。

専用言語のよさはプログラムが簡単に書けるという点にだけあるわけではない。その汎用性の高さもその特徴のひとつといえる。たとえばの話，前出の「割り込み」なども軽くこなす。モデル内部での細かな作業にも気を配らなければならないBASIC版に対し，モデルの構築だけに集中できるのが専用言語のいいところといえるだろう。

さて，いいことずくめの専用言語なのだが，残念ながらX68000上には存在していない。これでいいのだろうか。

NEXT

もちろんこれでいいわけがない。この連載ではX68000用のシミュレーション専用言語を紹介する予定である。おおそうか来月か，と焦ってはいけない。

来月は目先を変えて連続変化モデルを考えてみる予定である。できればはやりのファジーにも触れたいと思っているのだが。

とはいえども所詮「予定は未定」，すべては神の御心のままに。もっとも皆さんの意見によっては神様も心がわりするかもしれないので，思うところがあれば，どんどん意見してほしい。

ではまた，来月。

参考文献

・システムとシミュレーション，渡辺茂編，共立出版

・コンピュータシミュレーション，中西俊男，近代科学社

表2　リスト1の実行結果

```
Result

Simulation time (t) : 31812 s

 -- Facility --
  name        average          number       average
  or no.    utilization        entries    time / Xact
   1          0.9840            4458         7.023            in use
   2          0.9854            4482         6.995            in use
   3          0.2322            1062         6.957


 --  Queue  --
  no.       max      average       total     zero   percent    average    current
          contents   contents     entries  entries   zero    time /Xact  contents
   1        328      165.214       4780        4     0.001    1099.536      322
   2        328      165.198       4803        3     0.001    1094.166      321
   3          3        0.028       1062      826     0.778       0.831        0
```

表3　リスト2の実行結果

```
Result

Simulation time (t) : 29736 s

 -- Facility --
  name        average          number       average
  or no.    utilization        entries    time / Xact
   1          0.7599            3245         6.963
   2          0.7642            3251         6.992            in use
   3          0.8253            3506         7.001            in use


 --  Queue  --
  no.       max      average       total     zero   percent    average    current
          contents   contents     entries  entries   zero    time /Xact  contents
   1          2        0.160       3245     1960     0.604      1.467        0
   2          2        0.157       3252     1976     0.608      1.435        1
   3          4        0.278       3506     1643     0.469      2.358        0
```

リスト3

```
                    GPSS sample list

          BLOCK NO.           LOC     OPERATION        A,B,C,D,E,F,G

                              *
                              1       FUNCTION         RN(1),D2
                               0.50     1  1.00         2
                              *
                  1                   GENERATE         3,2
                  2                   TRANSFER         .100,JNR3,JNR1
                              *
                  3            JNR1   ASSIGN           1,3
                              *
                  4            JNR2   QUEUE            P1
                  5                   SEIZE            P1
                  6                   DEPART           P1
                  7                   ADVANCE          7,2
                  8                   RELEASE          P1
                  9                   TERMINATE        1
                              *
                 10            JNR3   ASSIGN           1,FN1
                 11                   GATE U           1,JNR5
                 12                   GATE U           2,JNR4
                 13                   TEST NE          Q1,Q2,JNR2
                 14                   TEST G           Q1,Q2,JNR6
                              *
                 15            JNR4   ASSIGN           1,2
                 16                   TRANSFER         ,JNR2
                              *
                 17            JNR5   GATE U           2,JNR2
                              *
                 18            JNR6   ASSIGN           1,1
                 19                   TRANSFER         ,JNR2
                              /       START            1000
                              /       END
```

THE SENTINEL

●S-OSとゲーム

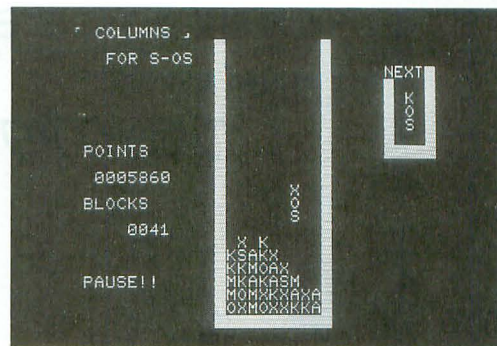
システム中心の堅い記事が多いと思われがちなS-OSですが,初期の頃からゲーム開発パッケージBEMSを始めとして意欲的にゲーム分野にも取り組んできました。BEMSはアクションゲーム用のパッケージでしたが提言という色合いが強く,実際にアプリケーションゲームが発表されることがなかったのが惜しまれるところです。S-OSのゲーム史はアドベンチャーゲームやシューティングゲームをちりばめながらも,パズルゲーム中心で流れてきたといってもいいでしょう。

開発の容易さから,TTCやSTACKなどで書かれたゲームの多かったなかで,今月は久し振りにアセンブラで書かれたアクションパズルゲームが登場します。しかもあのCOLUMNSとくれば,これは見逃す手はありません。X68000では付録ディスクで配布されたYETが有名ですが,S-OSユーザーとしては指をくわえて見ているしかありませんでした。タイルが降り積もっていく緊張感。次第に速くなるタイルの落下速度。連鎖反応を起こして一気に山が崩れるときの爽快感。どうぞ,このCOLUMNSで思う存分はまってください。

第102部
ブロックアクションゲームCOLUMNS

●COLUMNS登場

このCOLUMNSは作者の堀井さんが初めて作った100行以上のプログラムだそうです。タイルの落下速度は次第に速くなっていき,ある程度速くなると最初の速度に戻って再び速くなっていきます。タイルの回転方向はゲームセンターのCOLUMNSやYETとは逆になっていますので,戸惑う方がいらっしゃるかもしれません。また,タイルのキャラクタが気に入らないという方もいらっしゃることでしょう。本文でも触れられているように,簡単に自分好みのキャラクタを設定できますので,改造してみてはいかがでしょうか。個人的には,「ロ・＝＊＋■」などが判別しやすくていいのではないかと思います。

このプログラムは泉氏のX-BASIC調理実習で取り上げたYETのアルゴリズムを参考に作ったということです。Oh!Xの言語関係の連載ではさまざまなアルゴリズムが解説されていますので,自分でプログラムを作ってみるのに格好の教材となることでしょう。高級言語で書かれたプログラムをアセンブラに「移植」するのではなく,そのアルゴリズムを自分でアセンブラプログラムに直す。連載のこのような利用法も面白いのではないでしょうか。

●S-OSの系譜

1987年4月号では,S-OSの歴史に新しい1ページが加わりました。初のアクションゲームが登場したのです。それまでS-OSのゲームといえばパズルゲームが主流でした。これには理由がありました。誰もがS-OSの文字表示は遅いと考えていたのです。

S-OSは各機種のモニタやBIOSを利用して画面表示を行っています。しかも文字セットを統一するため,共通文字セットから各機種の文字セットへの変換を行ったあと,モニタ・BIOSを利用するのです。この2重の手間が介在したのでは,とてもアクションゲームを実現できるほどのスピードは望めない。スタッフを含め多くの人がそう思い込んでいたとしても不思議ではないでしょう。

事実,E-MATEなどのエディタでも,文字表示用の各機種独自のサブルーチンを組み込んで使うことを前提としていました。もっとも,これはE-MATEが必ず1行80字の文字を出力している,つまり1画面2000文字のキャラクタをスクロールのたびに書き込んでいるという隠れた理由もあります。いずれにせよ,S-OSの表示が遅いという実例となってしまったのは間違いないでしょう。

INVADER GAME, TANGERINEの2つのゲームは,これらの暗黙の思い込みを見事に打破してくれました。INVADER GAMEはその名のとおり古典的なインベーダーゲームですが,列をなしたインベーダーが下へ迫ってくる部分は結構重くなりそうな処理です。それに輪をかけて大変そうなのがTANGERINE。こちらは横スクロールシューティングゲームで,スクロールと自機・敵・弾の重ね合わせという重そうな処理を抱えています。作者の片岡氏は,S-OSには難しいといわれているアクションゲームに敢えて挑戦してみたということです。ところが共通ルーチンしか使っていないのに実際にはウェイトを入れなければゲームにならないほどの速度が出たとのこと。あえて不可能に挑戦してみるこの心意気が,新しいS-OS領域を生み出したのでした。

全機種共通S-OS“SWORD”要

ブロックアクションゲームCOLUMNS

“Beyond the columns” そして “Yet another columns”, COLUMNSの源流に忠実に，いまS-OS上にもCOLUMNSがやってきました（ゲーセン版とは関係ないですよ)。存分にプレイしてみてください。

Horii Toshikazu
堀井 俊和

S-OS “SWORD” 用の「COLUMNS」を発表します。このプログラムは1990年6月号で発表されたX68000用の「Yet Another Column」の移植版です。X68000版を見て指がうずいていたS-OSユーザーの皆さん，とうとうS-OSでCOLUMNSがプレイできるのです。

入力&実行方法

ダンプリストを打ち込まれる方は8000H～883EHをMACINTO-Cなどで入力し，その範囲でセーブしてください。実行アドレスは8000Hです。ソースで入力される方はエディタで打ち込んでREDAなどでアセンブルしてください。OFFSETがついているので，先にセーブしてロードしなおしてから実行してください。

ゲーム内容

もはや，多くの人が知っていると思いますが，念のため説明します。基本的な操作はテトリスなんかと同じです。上から落ちる3連のブロックを使って下に積もったブロックを消していきます。ブロックを消すには同じ文字のブロックを3個以上，縦・横・斜めに並べなければなりません。

キー操作は4，6で左右，5でブロックを入れ替える（3連ブロックの順序を下から上にずらす)，スペースキーでブロックを落とします。そのほか，Pでポーズ，シフトブレイクで中断です。テンキーのない機種では後述の変更点を参照してください。

得点は少々変わっており，落としたところ（自由落下でならブロックが止まったところ）が下から何段目かを調べ，それから3を引き（この時点で1以下だとすべて2になります)，それを2で割り，またそれに2をかけ，10倍したものがブロック1個の点数となります。これはプログラムの関係でなったことで別に意図はありません。

またスピードはゲーセン版テトリスのようになっておりどんどんスピードが上がっていって，最速を超えるとかなり遅くなってループを繰り返すというものです。

このゲーム，1巡するのは簡単ですが，2巡目はかなり難しいでしょうね。だいたい1巡を超えると半分以上ブロックが積まれていたりしますからね。でもやっぱり，ハマリますよ。

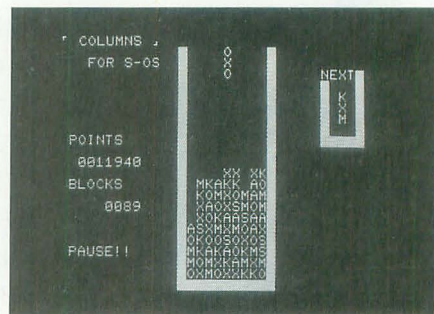
プログラムについて

はっきりいって，私はこのゲーム，わりとすんなり完成しました。というのも，泉大介氏のX-BASIC入門をかなり参考にさせていただいたからです。このゲームのネックとなるブロックのチェックルーチンから，ブロックを消して落下させるという一連の動作の大まかなアルゴリズムを使わせていただきました。本当に素晴らしいアルゴリズムでした。泉さんありがとうございます。もうひとつ，乱数発生ルーチンをOh! MZ1986年11月号から使わせていただきました。プログラムの構成についてはソースにもある程度注釈を入れておきますのでそれを参考にしてください。また，チェックルーチンの内容は前述の泉氏の記事（1990年6，7月号）を見てください。

プログラムの変更

これは各機種で違うスピードの調節やキー入力の変更などを行うものです。ダンプリストから入力した方はすべて打ち込んでCRCなどを確認したあと，ソースリストから入力した方は該当部分を書き換えて再アセンブルしてください。

まず，キー配置の変更から，

81D3H	左への移動
81D8H	右への移動
81DDH	ブロックの回転
81E2H	ブロックを落とす

に変更したい文字のキャラクタコードを書き込んでください。

スピード調節は何カ所にも分かれているのですが，ここではひとつだけ挙げておきます。あとは自由課題ということで。

速くするには80FFHと8104Hの2バイトを組み合わせた16進4桁の数値を小さくします（下限は0010Hまで)。逆に遅くするには先ほどの2バイトを増やせばいいのです。この場合は0200Hくらいまでにしたほうがいいでしょう。

最後にブロックキャラクタの変更です。デフォルトではA，O，X，M，K，Sといった比較的変わった形のキャラクタを集めていますが，変えたいという方はご自由にどうぞ。8528Hからの6バイトに使いたいキャラクタを書き込んでください。

＊　　＊　　＊

このプログラムは私にとって初めての100行以上のプログラムです。S-OSと参考文献のおかげで挫折もせずに完成できました。まあ，それほどたいしたモノではないですがね。でもやっぱり完成すると嬉しかったものです。はっきりいって，あまり綺麗なプログラムではありませんが，プログラム本体やワークエリアにも注釈を付けておいたので参考にしてください。

また，このゲームは入門用の題材としてはとてもいいと思いますので，できれば皆さんも一から作ってみてください。テトリスもどきのほうが簡単だけど，こっちのほうが面白いんじゃないでしょうか。

リスト1

```
8000 3E 0C CD F4 1F CD 30 20 : 47
8008 21 33 85 06 03 AF 77 23 : 2B
8010 10 FC 21 00 00 22 36 85 : 0A
8018 21 00 08 22 39 85 3E 00 : 47
8020 32 3E 88 06 15 21 0F 02 : 45
8028 CD 1E 20 CD E2 1F 7B 20 : 74
8030 20 20 20 20 20 20 20 20 : 00
8038 7B 00 24 10 EB CD 1E 20 : A5
8040 CD E2 1F 7B 7B 7B 7B 7B : 35
8048 7B 7B 7B 7B 7B 7B 00 21 : 03
8050 1F 04 CD 1E 20 CD E2 1F : FC
8058 4E 45 58 54 7B 00 06 05 : C5
8060 24 CD 1E 20 CD E2 1F 7B : 78
8068 20 20 20 7B 00 10 F1 24 : 00
8070 CD 1E 20 CD E2 1F 7B 7B : CF
8078 7B 7B 7B 00 21 00 01 CD : 60
-----------------------------------
SUM: 6B E3 FF EF BE 24 D2 D1 4879

8080 1E 20 CD E2 1F 20 20 A2 : EE
8088 20 43 4F 4C 55 4D 4E 53 : 41
8090 20 A3 0D 0D 20 20 20 20 : 5D
8098 20 46 4F 52 20 53 2D 4F : F6
80A0 53 00 21 03 0A CD 1E 20 : 8C
80A8 CD E2 1F 50 4F 49 4E 54 : 58
80B0 53 00 26 0E CD 1E 20 CD : 5F
80B8 E2 1F 42 4C 4F 43 4B 53 : BF
80C0 00 CD C4 1F CD 1C 81 CD : E7
80C8 D3 80 CD 9E 81 CD 63 81 : F0
80D0 C3 C7 81 21 24 85 11 21 : 07
80D8 85 01 03 00 ED B0 21 14 : 5B
80E0 04 22 1F 85 21 36 85 3E : E4
80E8 01 86 27 77 23 3E 00 8E : 14
80F0 77 2A 39 85 7D 2E 10 95 : AF
80F8 6F 7C 26 00 9C 67 FE 00 : 12
-----------------------------------
SUM: D9 B0 DA 99 E5 7E 3B DC 0E8E

8100 20 08 7D FE 50 30 03 21 : 47
8108 00 06 22 39 85 2A 1F 85 : B4
8110 CD 1B 20 FE 20 CA 1C 81 : 8D
8118 E1 C3 CD 84 11 24 85 06 : B5
8120 03 CD 32 81 21 28 85 85 : D6
8128 6F 7E 12 13 10 F3 CD 8A : 6C
8130 81 C9 D9 ED 5B 2E 85 ED : 0B
8138 4B 30 85 CD 50 81 22 2E : EE
8140 85 ED 5F 32 32 85 AC E6 : 4C
8148 07 FE 06 D2 33 81 D9 C9 : 33
8150 21 00 00 3E 10 29 CB 23 : 86
8158 CB 12 DA 5E 81 09 3D C2 : 9E
8160 55 81 C9 D9 2A 1F 85 11 : 57
8168 21 85 06 03 CD 1E 20 1A : D4
8170 CD F4 1F 25 13 10 F5 D9 : F6
8178 C9 D9 2A 1F 85 06 03 CD : 46
-----------------------------------
SUM: 90 00 85 C7 67 9D E6 BC 2A6A

8180 1E 20 CD F1 1F 25 10 F7 : 47
8188 D9 C9 21 21 08 11 24 85 : A6
8190 06 03 CD 1E 20 1A CD F4 : EF
8198 1F 25 13 10 F5 C9 21 04 : 4A
81A0 0C CD 1E 20 21 35 85 7E : 70
81A8 CD C1 1F 2B 7E CD C1 1F : 03
81B0 2B 7E CD C1 1F 3E 30 CD : 91
81B8 F4 1F 21 07 10 CD 1E 20 : 56
81C0 2A 36 85 CD BE 1F C9 ED : 45
81C8 4B 39 85 ED 43 3B 85 CD : C6
81D0 FC 81 FE 34 CA 0D 82 FE : 06
81D8 36 CA 1C 82 FE 35 CA 2B : C6
81E0 82 FE 20 CA 40 82 FE 50 : 7A
81E8 CA 70 82 FE 1B CA A1 82 : C2
81F0 ED 4B 3B 85 0B 78 B1 20 : 4C
81F8 D2 C3 BC 82 CD D0 1F 21 : B0
-----------------------------------
SUM: C6 72 B6 92 06 56 BF F4 24CC

8200 38 85 57 7E BA 20 03 3E : AD
8208 00 C9 7A 77 C9 2A 1F 85 : 51
8210 2D CD 1B 20 FE 20 C2 F0 : 05
8218 81 C3 B0 82 2A 1F 85 2C : 70
8220 CD 1B 20 FE 20 C2 F0 81 : 59
8228 C3 B0 82 21 21 85 7E 57 : 91
8230 23 7E 72 23 57 7E 72 2B : A8
8238 2B 77 CD 63 81 C3 F0 81 : 87
8240 2A 1F 85 44 3E 17 90 FE : F5
8248 05 30 02 3E 05 D6 03 CB : 1E
8250 3F 32 27 85 11 00 30 1B : 79
8258 7B B2 28 F8 24 CD 1B 20 : 79
8260 FE 20 C2 D4 82 CD 79 81 : FD
8268 22 1F 85 CD 63 81 18 E4 : 73
8270 E5 21 03 14 CD 1E 20 CD : F5
8278 E2 1F 50 41 55 53 45 21 : A0
-----------------------------------
SUM: 94 50 ED 31 43 8A 0D BA 04A1

8280 21 00 CD C4 1F CD FC 81 : 1B
8288 FE 20 20 F9 21 03 14 CD : 3C
8290 1E 20 CD E2 1F 20 20 20 : 6C
8298 20 20 20 20 00 E1 C3 F0 : 14
82A0 81 21 00 00 2B 7C B5 20 : 1E
82A8 FB 21 00 00 CD 1E 20 C9 : F0
82B0 CD 79 81 22 1F 85 CD 63 : BD
82B8 81 C3 F0 81 2A 1F 85 24 : A7
82C0 CD 1B 20 FE 20 C2 D9 82 : 43
82C8 CD 79 81 22 1F 85 CD 63 : BD
82D0 81 C3 C7 81 CD C4 1F 18 : 54
82D8 14 2A 1F 85 44 3E 17 90 : 0B
82E0 FE 05 30 02 3E 05 D6 03 : 51
82E8 CB 3F 32 27 85 21 3D 85 : CB
82F0 ED 5B 1F 85 06 03 7B 77 : E7
82F8 23 7A 77 23 15 10 F7 36 : 89
-----------------------------------
SUM: 2F 78 CA 59 CE 91 7B 90 EC12

8300 00 CD 0D 83 CD E8 83 CD : 62
8308 44 84 C3 01 83 11 B9 86 : 5F
8310 ED 53 33 88 11 3D 85 ED : BB
8318 53 B7 86 1A 6F 13 1A 67 : AD
8320 13 CD 1B 20 4F ED 53 B7 : 61
8328 86 CD 3B 83 ED 5B B7 86 : 96
8330 1A B7 20 E7 ED 5B 33 88 : DB
8338 AF 12 C9 E5 06 01 ED 5B : BE
8340 33 88 7D 12 13 7C 12 13 : FE
8348 24 CD 1B 20 B9 20 09 7D : 8B
8350 12 13 7C 12 13 04 18 F0 : D2
8358 78 FE 03 38 06 ED 53 33 : 2A
8360 88 18 04 ED 5B 33 88 E1 : 88
8368 E5 2D CD 1B 20 B9 28 F9 : F4
8370 06 00 2C CD 1B 20 B9 20 : 13
8378 09 7D 12 13 7C 12 13 04 : 50
-----------------------------------
SUM: 43 E6 EE F9 F6 98 07 78 658F

8380 18 F0 78 FE 03 38 06 ED : AC
8388 53 33 88 18 04 ED 5B 33 : A5
8390 88 E1 E5 25 2D CD 1B 20 : A8
8398 B9 28 F8 06 00 2C 24 CD : FC
83A0 1B 20 B9 20 09 7D 12 13 : BF
83A8 7C 12 13 04 18 EF 78 FE : 22
83B0 03 38 06 ED 53 33 88 18 : 54
83B8 04 ED 5B 33 88 E1 2C 25 : 39
83C0 CD 1B 20 B9 28 F8 06 00 : E7
83C8 2D 24 CD 1B 20 B9 20 09 : 3B
83D0 7D 12 13 7C 12 13 04 18 : 5F
83D8 EF 78 FE 03 38 05 ED 53 : E5
83E0 33 88 C9 ED 5B 33 88 C9 : 50
83E8 11 B9 86 1A B7 28 51 3E : D8
83F0 2A 06 02 08 1A B7 28 38 : 6B
83F8 6F 13 1A 67 13 CD 1E 20 : 21
-----------------------------------
SUM: 8D A6 73 4E 01 46 14 2E 239D

8400 08 CD F4 1F 08 D5 11 35 : 0B
8408 88 7D D6 10 83 5F 3E 00 : 0B
8410 8A 57 3E 01 12 E5 3A 27 : 78
8418 85 21 33 85 86 27 77 3E : C0
8420 00 23 8E 27 77 3E 00 23 : B0
8428 8E 27 77 E1 D1 08 18 C3 : C1
8430 21 00 50 2B 7D B4 20 FB : E8
8438 11 B9 86 3E 20 10 B4 C9 : 3B
8440 E1 C3 C7 80 11 3D 85 ED : AB
8448 53 B7 86 06 09 2E 10 11 : EE
8450 35 88 1A B7 C4 6D 84 2C : 6F
8458 13 10 F7 ED 5B B7 86 AF : 4E
8460 12 3A 3E 88 3D C0 32 3E : 7F
8468 88 CD C4 1F C9 AF 12 26 : E8
8470 16 E5 D9 E1 06 00 CD 1B : A3
8478 20 FE 20 28 04 04 25 18 : AB
-----------------------------------
SUM: AB C1 6F 00 51 4C C1 B4 FCB3

8480 F5 26 16 AF B0 28 03 25 : E0
8488 10 FD CD 1B 20 FE 20 C4 : F7
8490 9A 84 25 7C FE 01 20 F2 : D0
8498 D9 C9 E5 F5 24 CD 1B 20 : A8
84A0 FE 20 20 12 25 CD 1E 20 : 80
84A8 CD F1 1F 24 CD 1E 20 F1 : FD
84B0 F5 CD F4 1F 18 E6 ED 5B : 1B
84B8 B7 86 25 7D 12 13 7C 12 : 92
84C0 13 ED 53 B7 86 21 3E 88 : 77
84C8 36 01 F1 E1 C9 CD C4 1F : 82
84D0 CD C4 1F 21 10 0B CD 1E : D7
84D8 20 CD E2 1F 20 20 20 20 : 6E
84E0 20 20 20 20 20 00 24 CD : 91
84E8 1E 20 CD E2 1F 20 52 45 : C3
84F0 54 52 59 20 3F 20 00 24 : A2
84F8 CD 1E 20 CD E2 1F 20 20 : 19
-----------------------------------
SUM: 84 03 F0 D4 ED 50 8A B4 6068

8500 20 20 20 20 20 20 20 00 : E0
8508 CD D0 1F B7 28 FA FE 59 : EC
8510 CA 00 80 FE 4E 28 02 18 : D8
8518 EF 3E 0C CD F4 1F C9 00 : E2
8520 00 00 00 00 00 00 00 00 : 00
8528 41 4F 58 4D 4B 53 33 E9 : EF
8530 83 03 00 00 00 00 00 00 : 86
8538 00 00 00 00 00 00 00 00 : 00
8540 00 00 00 00 00 00 00 00 : 00
8548 00 00 00 00 00 00 00 00 : 00
8550 00 00 00 00 00 00 00 00 : 00
8558 00 00 00 00 00 00 00 00 : 00
8560 00 00 00 00 00 00 00 00 : 00
8568 00 00 00 00 00 00 00 00 : 00
8570 00 00 00 00 00 00 00 00 : 00
8578 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 6A 80 23 EF D5 B4 1C 5A 1DAD

8580 00 00 00 00 00 00 00 00 : 00
8588 00 00 00 00 00 00 00 00 : 00
8590 00 00 00 00 00 00 00 00 : 00
8598 00 00 00 00 00 00 00 00 : 00
85A0 00 00 00 00 00 00 00 00 : 00
85A8 00 00 00 00 00 00 00 00 : 00
85B0 00 00 00 00 00 00 00 00 : 00
85B8 00 00 00 00 00 00 00 00 : 00
85C0 00 00 00 00 00 00 00 00 : 00
85C8 00 00 00 00 00 00 00 00 : 00
85D0 00 00 00 00 00 00 00 00 : 00
85D8 00 00 00 00 00 00 00 00 : 00
85E0 00 00 00 00 00 00 00 00 : 00
85E8 00 00 00 00 00 00 00 00 : 00
85F0 00 00 00 00 00 00 00 00 : 00
85F8 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

8600 00 00 00 00 00 00 00 00 : 00
8608 00 00 00 00 00 00 00 00 : 00
8610 00 00 00 00 00 00 00 00 : 00
8618 00 00 00 00 00 00 00 00 : 00
8620 00 00 00 00 00 00 00 00 : 00
8628 00 00 00 00 00 00 00 00 : 00
8630 00 00 00 00 00 00 00 00 : 00
8638 00 00 00 00 00 00 00 00 : 00
8640 00 00 00 00 00 00 00 00 : 00
8648 00 00 00 00 00 00 00 00 : 00
8650 00 00 00 00 00 00 00 00 : 00
8658 00 00 00 00 00 00 00 00 : 00
8660 00 00 00 00 00 00 00 00 : 00
8668 00 00 00 00 00 00 00 00 : 00
8670 00 00 00 00 00 00 00 00 : 00
8678 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

8680 00 00 00 00 00 00 00 00 : 00
8688 00 00 00 00 00 00 00 00 : 00
8690 00 00 00 00 00 00 00 00 : 00
8698 00 00 00 00 00 00 00 00 : 00
86A0 00 00 00 00 00 00 00 00 : 00
86A8 00 00 00 00 00 00 00 00 : 00
86B0 00 00 00 00 00 00 00 00 : 00
86B8 00 00 00 00 00 00 00 00 : 00
86C0 00 00 00 00 00 00 00 00 : 00
86C8 00 00 00 00 00 00 00 00 : 00
86D0 00 00 00 00 00 00 00 00 : 00
86D8 00 00 00 00 00 00 00 00 : 00
86E0 00 00 00 00 00 00 00 00 : 00
86E8 00 00 00 00 00 00 00 00 : 00
86F0 00 00 00 00 00 00 00 00 : 00
86F8 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

8700 00 00 00 00 00 00 00 00 : 00
8708 00 00 00 00 00 00 00 00 : 00
8710 00 00 00 00 00 00 00 00 : 00
8718 00 00 00 00 00 00 00 00 : 00
8720 00 00 00 00 00 00 00 00 : 00
8728 00 00 00 00 00 00 00 00 : 00
8730 00 00 00 00 00 00 00 00 : 00
8738 00 00 00 00 00 00 00 00 : 00
8740 00 00 00 00 00 00 00 00 : 00
8748 00 00 00 00 00 00 00 00 : 00
8750 00 00 00 00 00 00 00 00 : 00
8758 00 00 00 00 00 00 00 00 : 00
8760 00 00 00 00 00 00 00 00 : 00
8768 00 00 00 00 00 00 00 00 : 00
8770 00 00 00 00 00 00 00 00 : 00
8778 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

8780 00 00 00 00 00 00 00 00 : 00
8788 00 00 00 00 00 00 00 00 : 00
8790 00 00 00 00 00 00 00 00 : 00
8798 00 00 00 00 00 00 00 00 : 00
87A0 00 00 00 00 00 00 00 00 : 00
87A8 00 00 00 00 00 00 00 00 : 00
87B0 00 00 00 00 00 00 00 00 : 00
87B8 00 00 00 00 00 00 00 00 : 00
87C0 00 00 00 00 00 00 00 00 : 00
87C8 00 00 00 00 00 00 00 00 : 00
87D0 00 00 00 00 00 00 00 00 : 00
87D8 00 00 00 00 00 00 00 00 : 00
87E0 00 00 00 00 00 00 00 00 : 00
87E8 00 00 00 00 00 00 00 00 : 00
87F0 00 00 00 00 00 00 00 00 : 00
87F8 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

8800 00 00 00 00 00 00 00 00 : 00
8808 00 00 00 00 00 00 00 00 : 00
8810 00 00 00 00 00 00 00 00 : 00
8818 00 00 00 00 00 00 00 00 : 00
8820 00 00 00 00 00 00 00 00 : 00
8828 00 00 00 00 00 00 00 00 : 00
8830 00 00 00 00 00 00 00 00 : 00
8838 00 00 00 00 00 00 00    : 00
-----------------------------------
SUM: 00 00 00 00 00 00 00 00 0000
```

↑うわー！　疲れてたんです。ごめんなさい(U)

リスト2

```
0000                1 ;**********************************************************
0000                2 ;*                                                        *
0000                3 ;*            『 COLUMNS 』         for S-OS              *
0000                4 ;*                                                        *
0000                5 ;**********************************************************
0000                6 ;
0000                7 ;
0000                8          OFFSET  0B000H-8000H
8000                9          ORG     8000H
8000               10 ;
8000               11 #PRINT   EQU     1FF4H
8000               12 #LOC     EQU     201EH
8000               13 #WIDCH   EQU     2030H
8000               14 #MPRINT  EQU     1FE2H
8000               15 #PRTHL   EQU     1FBEH
8000               16 #GETKY   EQU     1FD0H
8000               17 #SCRN    EQU     201BH
8000               18 #PRINTS  EQU     1FF1H
8000               19 #PRTHX   EQU     1FC1H
8000               20 #BELL    EQU     1FC4H
8000               21 ;
8000               22 ;
8000               23 START:                            ;SHOKI SETTEI
8000 3E 0C         24          LD      A,0CH            ;CLEAR
8002 CD F4 1F      25          CALL    #PRINT
8005 CD 30 20      26          CALL    #WIDCH           ;40*25
8008               27 WKCLR:
8008 21 33 85      28          LD      HL,POINTS        ;WORK AREA SHOKIKA
800B 06 03         29          LD      B,3              ;SCORE SHOKIKA
800D AF            30          XOR     A
800E               31 WKCLRLP:
800E 77            32          LD      (HL),A
800F 23            33          INC     HL
8010 10 FC         34          DJNZ    WKCLRLP
8012 21 00 00      35          LD      HL,0             ;BLOCK SUU SHOKIKA
8015 22 36 85      36          LD      (BLKS),HL
8018 21 00 08      37          LD      HL,0800H         ;SPEED SHOKIKA
801B 22 39 85      38          LD      (SPEED),HL
801E 3E 00         39          LD      A,0              ;BELL WORK(WORK SANSHOU)
8020 32 3E 88      40          LD      (BELLWK),A       ;      SHOKIKA
8023               41          ;
8023               42 MKSCRN:                           ;GAMEN SAKUSEI
8023 06 15         43          LD      B,21
8025 21 0F 02      44          LD      HL,020FH
8028 CD 1E 20      45 MKSCLP:  CALL    #LOC             ;WAKU SAKUSEI
802B CD E2 1F      46          CALL    #MPRINT
802E 7B            47          DB      7BH
802F 20 20 20      48          DM      "         "
8032 20 20 20
8035 20 20 20
8038 7B            49          DB      7BH
8039 00            50          DB      00H
803A 24            51          INC     H
803B 10 EB         52          DJNZ    MKSCLP
803D CD 1E 20      53          CALL    #LOC
8040 CD E2 1F      54          CALL    #MPRINT
8043 7B 7B 7B      55          DB      7BH,7BH,7BH,7BH,7BH,7BH,7BH,7BH
8046 7B 7B 7B
8049 7B 7B
804B 7B 7B 7B      56          DB      7BH,7BH,7BH,00H
804E 00
804F               57          ;
804F 21 1F 04      58          LD      HL,041FH         ;NEXT BLOCK WINDOW
8052 CD 1E 20      59          CALL    #LOC             ;      NO SAKUSEI
8055 CD E2 1F      60          CALL    #MPRINT
8058 4E 45 58      61          DM      "NEXT"
805B 54
805C 7B 00         62          DB      7BH,00H
805E 06 05         63          LD      B,5
8060 24            64 NSCLP:   INC     H
8061 CD 1E 20      65          CALL    #LOC
8064 CD E2 1F      66          CALL    #MPRINT
8067 7B 20 20      67          DB      7BH,20H,20H,20H,7BH,00H
806A 20 7B 00
806D 10 F1         68          DJNZ    NSCLP
806F 24            69          INC     H
8070 CD 1E 20      70          CALL    #LOC
8073 CD E2 1F      71          CALL    #MPRINT
8076 7B 7B 7B      72          DB      7BH,7BH,7BH,7BH,7BH,00H
8079 7B 7B 00
807C               73          ;
807C 21 00 01      74          LD      HL,0100H         ;SONOTA
807F CD 1E 20      75          CALL    #LOC
8082 CD E2 1F      76          CALL    #MPRINT
8085 20 20 A2      77          DM      "  『 COLUMNS 』"
8088 20 43 4F
808B 4C 55 4D
808E 4E 53 20
8091 A3
8092 0D 0D         78          DB      0DH,0DH
8094 20 20 20      79          DM      "     FOR S-OS"
8097 20 20 46
809A 4F 52 20
809D 53 2D 4F
80A0 53
80A1 00            80          DB      00H
80A2               81          ;
80A2 21 03 0A      82          LD      HL,0A03H
80A5 CD 1E 20      83          CALL    #LOC
80A8 CD E2 1F      84          CALL    #MPRINT
80AB 50 4F 49      85          DM      "POINTS"
80AE 4E 54 53
80B1 00            86          DB      00H
80B2 26 0E         87          LD      H,0EH
80B4 CD 1E 20      88          CALL    #LOC
80B7 CD E2 1F      89          CALL    #MPRINT
80BA 42 4C 4F      90          DM      "BLOCKS"
80BD 43 4B 53
80C0 00            91          DB      00H
80C1               92 ;
80C1 CD C4 1F      93          CALL    #BELL            ;HAJIMEDAKE BELL
80C4 CD 1C 81      94          CALL    BLKGN2           ;HITOTUME NO BLOCK
80C7               95 ;
80C7               96 ;
80C7               97 MAIN:
80C7 CD D3 80      98          CALL    BLKGEN           ;BLOCK SAKUSEI
80CA CD 9E 81      99          CALL    SCRPRT           ;SCORE NO PRINT
80CD CD 63 81     100          CALL    BLKPRT           ;HAJIMENI BLOCK WO HYOUJI
80D0 C3 C7 81     101          JP      BLKMOVE          ;IDOU SONOTA
80D3              102 ;
80D3              103 ;
80D3              104 BLKGEN:
80D3 21 24 85     105          LD      HL,NXTBLK        ;HL=NEXT BLOCK ADRESS
80D6 11 21 85     106          LD      DE,NOWBLK        ;DE=IMANO BLOCK ADRESS
80D9 01 03 00     107          LD      BC,3             ;BLOCK HA 3 KO
80DC ED B0        108          LDIR                     ;NEXT WO NOW NI TENSOU
80DE 21 14 04     109          LD      HL,0414H         ;ZAHYOU DAINYU
80E1 22 1F 85     110          LD      (BLKXY),HL
80E4 21 36 85     111          LD      HL,BLKS          ;BLOCK SUU +1
80E7 3E 01        112          LD      A,1
80E9 86           113          ADD     A,(HL)
80EA 27           114          DAA
80EB 77           115          LD      (HL),A
80EC 23           116          INC     HL
80ED 3E 00        117          LD      A,0
80EF 8E           118          ADC     A,(HL)
80F0 77           119          LD      (HL),A
80F1 2A 39 85     120          LD      HL,(SPEED)       ;SPEED UP
80F4 7D           121          LD      A,L
80F5 2E 10        122          LD      L,10H            ;SPEED UP NO RITSU
80F7 95           123          SUB     L
80F8 6F           124          LD      L,A
80F9 7C           125          LD      A,H
80FA 26 00        126          LD      H,0
80FC 9C           127          SBC     A,H
80FD 67           128          LD      H,A
80FE FE 00        129          CP      00H              ;HL=0050H IKA NARA
8100 20 08        130          JR      NZ,GNLP          ;      HL=0600H NI
8102 7D           131          LD      A,L              ;      MODOSU
8103 FE 50        132          CP      50H
8105 30 03        133          JR      NC,GNLP
8107 21 00 06     134          LD      HL,0600H
810A              135 GNLP:
810A 22 39 85     136          LD      (SPEED),HL
810D 2A 1F 85     137          LD      HL,(BLKXY)       ;SYUTSUGEN BASYO NI
8110 CD 1B 20     138          CALL    #SCRN            ;      BLOCK ARI?
8113 FE 20        139          CP      ' '
8115 CA 1C 81     140          JP      Z,BLKGN2
8118 E1           141          POP     HL
8119 C3 CD 84     142          JP      GMOVR
811C              143 BLKGN2:
811C 11 24 85     144          LD      DE,NXTBLK        ;NEXT BLOCK NO SAKUSEI
811F 06 03        145          LD      B,3
8121 CD 32 81     146 BLGLP:   CALL    RANDOM           ;RANSUU HASSEI
8124 21 28 85     147          LD      HL,CHRTBL        ;1-6 NO RANSUU WO
8127 85           148          ADD     A,L              ;      CHARACTER NI
8128 6F           149          LD      L,A              ;      HENKAN
8129 7E           150          LD      A,(HL)
812A 12           151          LD      (DE),A
812B 13           152          INC     DE
812C 10 F3        153          DJNZ    BLGLP
812E CD 8A 81     154          CALL    NXTPRT
8131 C9           155          RET
8132              156          ;
8132              157 RANDOM:                           ;By Oh!MZ (SEP.1985)
8132 D9           158          EXX                      ;      SHITSUMON BAKO
8133 ED 5B 2E     159 RNDLP:   LD      DE,(OLDRND)      ;      YORI
8136 85
8137 ED 4B 30     160          LD      BC,(STEP)
813A 85
813B CD 50 81     161          CALL    MULTI
813E 22 2E 85     162          LD      (OLDRND),HL
8141 ED 5F        163          LD      A,R
8143 32 32 85     164          LD      (REFR),A
8146 AC           165          XOR     H
8147 E6 07        166          AND     07H              ;7 IKA NO KAZU NI SURU
8149 FE 06        167          CP      6                ;5 IKA?
814B D2 33 81     168          JP      NC,RNDLP
814E D9           169          EXX
814F C9           170          RET
8150              171          ;
8150              172 MULTI:
8150 21 00 00     173          LD      HL,0000H
8153 3E 10        174          LD      A,10H
8155 29           175 MLOOP:   ADD     HL,HL
8156 CB 23        176          SLA     E
8158 CB 12        177          RL      D
815A DA 5E 81     178          JP      C,SKIP
815D 09           179          ADD     HL,BC
815E 3D           180 SKIP:    DEC     A
815F C2 55 81     181          JP      NZ,MLOOP
8162 C9           182          RET
8163              183 ;
8163              184 ;
8163              185 BLKPRT:                           ;BLOCK HYOUJI
8163 D9           186          EXX
8164 2A 1F 85     187          LD      HL,(BLKXY)
8167 11 21 85     188          LD      DE,NOWBLK
816A 06 03        189          LD      B,3
816C CD 1E 20     190 BLPRLP:  CALL    #LOC
816F 1A           191          LD      A,(DE)
8170 CD F4 1F     192          CALL    #PRINT
8173 25           193          DEC     H
8174 13           194          INC     DE
8175 10 F5        195          DJNZ    BLPRLP
8177 D9           196          EXX
8178 C9           197          RET
8179              198          ;
8179              199 PRTSP:                            ;BLOCK SHOUKYO
8179 D9           200          EXX
817A 2A 1F 85     201          LD      HL,(BLKXY)
817D 06 03        202          LD      B,3
817F CD 1E 20     203 PRTSLP:  CALL    #LOC
8182 CD F1 1F     204          CALL    #PRINTS
8185 25           205          DEC     H
8186 10 F7        206          DJNZ    PRTSLP
8188 D9           207          EXX
8189 C9           208          RET
818A              209          ;
818A              210 NXTPRT:                           ;NEXT WINDOW NI
818A 21 21 08     211          LD      HL,0821H         ;      NEXT BLOCK
818D 11 24 85     212          LD      DE,NXTBLK        ;      WO HYOUJI
8190 06 03        213          LD      B,3
8192              214 NTPRLP:
8192 CD 1E 20     215          CALL    #LOC
8195 1A           216          LD      A,(DE)
8196 CD F4 1F     217          CALL    #PRINT
8199 25           218          DEC     H
819A 13           219          INC     DE
819B 10 F5        220          DJNZ    NTPRLP
819D C9           221          RET
819E              222 ;
819E              223 ;
819E              224 SCRPRT:                           ;SCORE NO PRINT
819E 21 04 0C     225          LD      HL,0C04H
81A1 CD 1E 20     226          CALL    #LOC
81A4 21 35 85     227          LD      HL,POINTS+2      ;JYOUI KARA
81A7 7E           228          LD      A,(HL)
81A8 CD C1 1F     229          CALL    #PRTHX
81AB 2B           230          DEC     HL
81AC 7E           231          LD      A,(HL)
81AD CD C1 1F     232          CALL    #PRTHX
81B0 2B           233          DEC     HL
81B1 7E           234          LD      A,(HL)
81B2 CD C1 1F     235          CALL    #PRTHX
81B5 3E 30        236          LD      A,'0'            ;SCORE WO 10 BAI
81B7 CD F4 1F     237          CALL    #PRINT
81BA 21 07 10     238          LD      HL,1007H
81BD CD 1E 20     239          CALL    #LOC
81C0 2A 36 85     240          LD      HL,(BLKS)        ;BLOCK SUU PRINT
81C3 CD BE 1F     241          CALL    #PRTHL
81C6 C9           242          RET
81C7              243 ;
81C7              244 ;
81C7              245 BLKMOVE:
81C7 ED 4B 39     246          LD      BC,(SPEED)       ;BLOCK RAKKA NO SPEED
81CA 85
81CB              247 KEYLP:
81CB ED 43 3B     248          LD      (SPDWK),BC       ;SPEED COUNTER
81CE 85
81CF              249                                   ;      WO PUSH
81CF CD FC 81     250          CALL    INKEY
81D2 FE 34        251          CP      '4'
81D4 CA 0D 82     252          JP      Z,LEFT
81D7 FE 36        253          CP      '6'
81D9 CA 1C 82     254          JP      Z,RIGHT
```


▶先日，1989年６月号を読み返し，その内容がいまでも新鮮なのには驚きました。今後も「いつ読んでも新鮮な」誌面づくりにがんばってください。　吉竹　宏志(17)大阪府

```
81DC FE 35         255          CP      '5'
81DE CA 2B 82      256          JP      Z,CHENGE
81E1 FE 20         257          CP      ' '
81E3 CA 40 82      258          JP      Z,FALL
81E6 FE 50         259          CP      'P'
81E8 CA 70 82      260          JP      Z,PAUSE
81EB FE 1B         261          CP      1BH
81ED CA A1 82      262          JP      Z,BREAK
81F0               263 KYLP2:
81F0 ED 4B 3B      264          LD      BC,(SPDWK)      ;SPEED COUNTER WO POP
81F3 85
81F4 0B            265 KYLP3:   DEC     BC              ;SPD CNTR HERASU
81F5 78            266          LD      A,B             ;ZERO NARA BLOCK RAKKA
81F6 B1            267          OR      C
81F7 20 D2         268          JR      NZ,KEYLP
81F9 C3 BC 82      269          JP      DOWN
81FC               270 INKEY:
81FC CD D0 1F      271          CALL    #GETKY          ;NON REPEAT NO GETKY
81FF 21 38 85      272          LD      HL,KEYBF        ;MAE NI OSHITA KEY
8202 57            273          LD      D,A
8203 7E            274          LD      A,(HL)
8204 BA            275          CP      D
8205 20 03         276          JR      NZ,INKY2
8207 3E 00         277          LD      A,0
8209 C9            278          RET
820A 7A            279 INKY2:   LD      A,D
820B 77            280          LD      (HL),A
820C C9            281          RET
820D               282          ;
820D               283 LEFT:                            ;HIDARI IDOU
820D 2A 1F 85      284          LD      HL,(BLKXY)
8210 2D            285          DEC     L
8211 CD 1B 20      286          CALL    #SCRN
8214 FE 20         287          CP      ' '
8216 C2 F0 81      288          JP      NZ,KYLP2
8219 C3 B0 82      289          JP      MVATR           ;JP MOVE AFTER
821C               290          ;
821C               291 RIGHT:                           ;MIGI IDOU
821C 2A 1F 85      292          LD      HL,(BLKXY)
821F 2C            293          INC     L
8220 CD 1B 20      294          CALL    #SCRN
8223 FE 20         295          CP      ' '
8225 C2 F0 81      296          JP      NZ,KYLP2
8228 C3 B0 82      297          JP      MVATR
822B               298          ;
822B               299 CHENGE:                          ;BLOCK CHENGE
822B 21 21 85      300          LD      HL,NOWBLK       ;WORK NO NAIYOU WO
822E 7E            301          LD      A,(HL)          ;       IREKAERU
822F 57            302          LD      D,A
8230 23            303          INC     HL
8231 7E            304          LD      A,(HL)
8232 72            305          LD      (HL),D
8233 23            306          INC     HL
8234 57            307          LD      D,A
8235 7E            308          LD      A,(HL)
8236 72            309          LD      (HL),D
8237 2B            310          DEC     HL
8238 2B            311          DEC     HL
8239 77            312          LD      (HL),A
823A CD 63 81      313          CALL    BLKPRT          ;BLOCK WO HYOUJI
823D C3 F0 81      314          JP      KYLP2
8240               315          ;
8240               316 FALL:                            ;BLOCK WO OTOSU
8240 2A 1F 85      317          LD      HL,(BLKXY)      ;SITA KARA
8243 44            318          LD      B,H             ;       NANKOME DE
8244 3E 17         319          LD      A,23            ;       OTOSHITAKA?
8246 90            320          SUB     B
8247 FE 05         321          CP      5
8249 30 02         322          JR      NC,FALL0
824B 3E 05         323          LD      A,5
824D D6 03         324 FALL0:   SUB     3
824F CB 3F         325          SRL     A               ;1/2 NI SURU
8251 32 27 85      326          LD      (HEIGHT),A
8254               327 FALLP:
8254 11 00 30      328          LD      DE,3000H        ;KARA LOOP
8257 1B            329          DEC     DE
8258 7B            330          LD      A,E
8259 B2            331          OR      D
825A 28 F8         332          JR      Z,FALLP
825C 24            333          INC     H               ;DONDON OCHIRU
825D CD 1B 20      334          CALL    #SCRN
8260 FE 20         335          CP      20H
8262 C2 D4 82      336          JP      NZ,CHECK1
8265 CD 79 81      337          CALL    PRTSP
8268 22 1F 85      338          LD      (BLKXY),HL
826B CD 63 81      339          CALL    BLKPRT
826E 18 E4         340          JR      FALLP
8270               341          ;
8270               342 PAUSE:                           ;PAUSE KINOU
8270 E5            343          PUSH    HL
8271 21 03 14      344          LD      HL,1403H
8274 CD 1E 20      345          CALL    #LOC
8277 CD E2 1F      346          CALL    #MPRINT
827A 50 41 55      347          DM      "PAUSE!!"
827D 53 45 21
8280 21
8281 00            348          DB      0
8282 CD C4 1F      349          CALL    #BELL
8285 CD FC 81      350 PSELP:   CALL    INKEY           ;SPACE KEY DE SAIKAI
8288 FE 20         351          CP      ' '
828A 20 F9         352          JR      NZ,PSELP
828C 21 03 14      353          LD      HL,1403H
828F CD 1E 20      354          CALL    #LOC
8292 CD E2 1F      355          CALL    #MPRINT
8295 20 20 20      356          DM      "       "
8298 20 20 20
829B 20
829C 00            357          DB      0
829D E1            358          POP     HL
829E C3 F0 81      359          JP      KYLP2
82A1               360          ;
82A1               361 BREAK:                           ;TOCHUU NO BREAK
82A1 21 00 00      362          LD      HL,0
82A4 2B            363 BRKLP:   DEC     HL
82A5 7C            364          LD      A,H
82A6 B5            365          OR      L
82A7 20 FB         366          JR      NZ,BRKLP
82A9 21 00 00      367          LD      HL,0
82AC CD 1E 20      368          CALL    #LOC
82AF C9            369          RET
82B0               370          ;
82B0               371 MVATR:                           ;IDOU ATO NO SHORI
82B0 CD 79 81      372          CALL    PRTSP           ;BLOCK WO KESU
82B3 22 1F 85      373          LD      (BLKXY),HL
82B6 CD 63 81      374          CALL    BLKPRT          ;BLOCK WO KAKU
82B9 C3 F0 81      375          JP      KYLP2
82BC               376 ;
82BC               377 ;
82BC               378 DOWN:                            ;BLOCK NO RAKKA
82BC 2A 1F 85      379          LD      HL,(BLKXY)
82BF 24            380          INC     H
82C0 CD 1B 20      381          CALL    #SCRN
82C3 FE 20         382          CP      ' '             ;SHITANI NANIKA AREBA
82C5 C2 D9 82      383          JP      NZ,CHECK2       ;       CHECK ROOTINE
82C8 CD 79 81      384          CALL    PRTSP
82CB 22 1F 85      385          LD      (BLKXY),HL
82CE CD 63 81      386          CALL    BLKPRT
82D1 C3 C7 81      387          JP      BLKMOVE
82D4               388 ;
82D4               389 ;
82D4               390 CHECK1:                          ;SPACE KEY DENO RAKKA
82D4 CD C4 1F      391          CALL    #BELL           ;BELL ARI
82D7 18 14         392          JR      CHECK3
82D9               393 CHECK2:                          ;TADA NO RAKKA
82D9 2A 1F 85      394          LD      HL,(BLKXY)      ;SITA KARA
82DC 44            395          LD      B,H             ;       NANKOME DE
82DD 3E 17         396          LD      A,23            ;       OTOSHITAKA?
82DF 90            397          SUB     B
82E0 FE 05         398          CP      5
82E2 30 02         399          JR      NC,CHECK2L
82E4 3E 05         400          LD      A,5
82E6 D6 03         401 CHECK2L: SUB     3
82E8 CB 3F         402          SRL     A               ;1/2 NI SURU
82EA 32 27 85      403          LD      (HEIGHT),A
82ED               404 CHECK3:
82ED 21 3D 85      405          LD      HL,CHKWK        ;CHECK NO WORK NI
82F0 ED 5B 1F      406          LD      DE,(BLKXY)      ;     3 KO NO BLOCK
82F3 85
82F4               407 CHK3LP:                          ;     WO DAINYU
82F4 06 03         408          LD      B,3
82F6               409 CHK3LP2:
82F6 7B            410          LD      A,E
82F7 77            411          LD      (HL),A
82F8 23            412          INC     HL
82F9 7A            413          LD      A,D
82FA 77            414          LD      (HL),A
82FB 23            415          INC     HL
82FC 15            416          DEC     D
82FD 10 F7         417          DJNZ    CHK3LP2
82FF 36 00         418          LD      (HL),00H        ;END CODE=00H
8301               419 CHECK4:
8301 CD 0D 83      420          CALL    BLKCHK          ;KESERUKA?
8304 CD E8 83      421          CALL    CLEAR           ;KESU
8307 CD 44 84      422          CALL    ALFALL          ;OTOSU
830A C3 01 83      423          JP      CHECK4          ;MATA CHECK
830D               424 ;
830D               425 ;
830D               426 BLKCHK:
830D 11 B9 86      427          LD      DE,CLRWK
8310 ED 53 33      428          LD      (CLRWK2),DE
8313 88
8314 11 3D 85      429          LD      DE,CHKWK
8317 ED 53 B7      430          LD      (CHKWK2),DE
831A 86
831B               431 BLKCHK2:
831B 1A            432          LD      A,(DE)          ;1 KO ME NO BLOCK NO
831C 6F            433          LD      L,A             ;       ZAHYOU WO
831D 13            434          INC     DE              ;       DAINYUU
831E 1A            435          LD      A,(DE)
831F 67            436          LD      H,A
8320 13            437          INC     DE              ;TSUGINO BLOCK
8321 CD 1B 20      438          CALL    #SCRN
8324 4F            439          LD      C,A
8325 ED 53 B7      440          LD      (CHKWK2),DE     ;PUSH
8328 86
8329 CD 3B 83      441          CALL    TYNCHECK        ;TATE YOKO NANAME CHECK
832C ED 5B B7      442          LD      DE,(CHKWK2)     ;POP
832F 86
8330 1A            443          LD      A,(DE)          ;OWARIKA?
8331 B7            444          OR      A               ;       (DATA GA 0 KA?)
8332 20 E7         445          JR      NZ,BLKCHK2
8334 ED 5B 33      446          LD      DE,(CLRWK2)
8337 88
8338 AF            447          XOR     A
8339 12            448          LD      (DE),A
833A C9            449          RET
833B               450          ;
833B               451 TYNCHECK:
833B               452 TCHECK:                          ;TATENO CHECK
833B E5            453          PUSH    HL
833C 06 01         454          LD      B,1
833E ED 5B 33      455          LD      DE,(CLRWK2)     ;ICHIOU CLEAR WORK NI
8341 88
8342 7D            456          LD      A,L             ;       ZAHYOU WO
8343 12            457          LD      (DE),A          ;       IRERU
8344 13            458          INC     DE              ;(3 IJOU NAKEREBA
8345 7C            459          LD      A,H             ;       CLRWK WO
8346 12            460          LD      (DE),A          ;       KOUSHIN SHINAI)
8347 13            461          INC     DE
8348               462 TCHKLP:
8348 24            463          INC     H
8349 CD 1B 20      464          CALL    #SCRN
834C B9            465          CP      C               ;C=CHECK SITERU BLOCK
834D 20 09         466          JR      NZ,TCHKLP2
834F 7D            467          LD      A,L             ;ICHIOU DAINYUU
8350 12            468          LD      (DE),A
8351 13            469          INC     DE
8352 7C            470          LD      A,H
8353 12            471          LD      (DE),A
8354 13            472          INC     DE
8355 04            473          INC     B               ;B=IKUTSU NARABUKA
8356 18 F0         474          JR      TCHKLP
8358               475 TCHKLP2:
8358 78            476          LD      A,B             ;3 IJOU NARABU?
8359 FE 03         477          CP      3
835B 38 06         478          JR      C,TCHKLP3
835D ED 53 33      479          LD      (CLRWK2),DE     ;CLRWK KOUSHIN
8360 88
8361 18 04         480          JR      YCHECK
8363               481 TCHKLP3:
8363 ED 5B 33      482          LD      DE,(CLRWK2)     ;KOUSHIN SHINAI
8366 88
8367               483          ;
8367               484 YCHECK:                          ;YOKO CHECK
8367 E1            485          POP     HL
8368 E5            486          PUSH    HL
8369               487 YCHKLP:
8369 2D            488          DEC     L
836A CD 1B 20      489          CALL    #SCRN
836D B9            490          CP      C               ;C=CHECK SHITERU BLOCK
836E 28 F9         491          JR      Z,YCHKLP
8370               492 YCHKLP2:
8370 06 00         493          LD      B,0
8372               494 YCHKLP3:
8372 2C            495          INC     L               ;HITOTSU MODOSU
8373 CD 1B 20      496          CALL    #SCRN
8376 B9            497          CP      C
8377 20 09         498          JR      NZ,YCHKLP4
8379 7D            499          LD      A,L             ;ICHIOU DAINYUU
837A 12            500          LD      (DE),A
837B 13            501          INC     DE
837C 7C            502          LD      A,H
837D 12            503          LD      (DE),A
837E 13            504          INC     DE
837F 04            505          INC     B
8380 18 F0         506          JR      YCHKLP3
8382               507 YCHKLP4:
8382 78            508          LD      A,B             ;3 KO IJOU NARABU?
8383 FE 03         509          CP      3
8385 38 06         510          JR      C,YCHKLP5
8387 ED 53 33      511          LD      (CLRWK2),DE     ;NARABUNARA KOUSHIN
838A 88
838B 18 04         512          JR      LNCHECK
838D               513 YCHKLP5:
838D ED 5B 33      514          LD      DE,(CLRWK2)     ;KOUSHIN SHINAI
8390 88
8391               515          ;
8391               516 LNCHECK:                         ;HIDARI NANAME CHECK
8391 E1            517          POP     HL
8392 E5            518          PUSH    HL
8393               519 LNCHKLP:
8393 25            520          DEC     H
8394 2D            521          DEC     L
8395 CD 1B 20      522          CALL    #SCRN
```

THE SENTINEL

▶Oh!X編集部で締め切りに追われながら必死に原稿を書いている夢を見てしまった。これはいったい……。
伊藤　ヨウジ(19)茨城県

```
8398 B9        523          CP      C               ;C=CHECK SHITERU BLOCK
8399 28 F8     524          JR      Z,LNCHKLP
839B           525 LNCHKLP2:
839B 06 00     526          LD      B,0
839D           527 LNCHKLP3:
839D 2C        528          INC     L               ;IKISUGI WO MODOSU
839E 24        529          INC     H
839F CD 1B 20  530          CALL    #SCRN
83A2 B9        531          CP      C
83A3 20 09     532          JR      NZ,LNCHKLP4
83A5 7D        533          LD      A,L             ;ICHIOU DAINYUU
83A6 12        534          LD      (DE),A
83A7 13        535          INC     DE
83A8 7C        536          LD      A,H
83A9 12        537          LD      (DE),A
83AA 13        538          INC     DE
83AB 04        539          INC     B
83AC 18 EF     540          JR      LNCHKLP3
83AE           541 LNCHKLP4:
83AE 78        542          LD      A,B             ;3 IJOU?
83AF FE 03     543          CP      3
83B1 38 06     544          JR      C,LNCHKLP5
83B3 ED 53 33  545          LD      (CLRWK2),DE     ;KOUSHIN
83B6 88
83B7 18 04     546          JR      RNCHECK
83B9           547 LNCHKLP5:
83B9 ED 5B 33  548          LD      DE,(CLRWK2)     ;KOUSHIN SHINAI
83BC 88
83BD           549          ;
83BD           550 RNCHECK:                         ;MIGI NANAME CHECK
83BD E1        551          POP     HL
83BE           552 RNCHKLP:
83BE 2C        553          INC     L
83BF 25        554          DEC     H
83C0 CD 1B 20  555          CALL    #SCRN
83C3 B9        556          CP      C               ;C=CHECK SHITERU BLOCK
83C4 28 F8     557          JR      Z,RNCHKLP
83C6           558 RNCHKLP2:
83C6 06 00     559          LD      B,0
83C8           560 RNCHKLP3:
83C8 2D        561          DEC     L               ;IKISUGI WO MODOSU
83C9 24        562          INC     H
83CA CD 1B 20  563          CALL    #SCRN
83CD B9        564          CP      C
83CE 20 09     565          JR      NZ,RNCHKLP4
83D0 7D        566          LD      A,L             ;ICHIOU DAINYUU
83D1 12        567          LD      (DE),A
83D2 13        568          INC     DE
83D3 7C        569          LD      A,H
83D4 12        570          LD      (DE),A
83D5 13        571          INC     DE
83D6 04        572          INC     B
83D7 18 EF     573          JR      RNCHKLP3
83D9           574 RNCHKLP4:
83D9 78        575          LD      A,B             ;3 IJOU?
83DA FE 03     576          CP      3
83DC 38 05     577          JR      C,RNCHKLP5
83DE ED 53 33  578          LD      (CLRWK2),DE     ;KOUSHIN
83E1 88
83E2 C9        579          RET
83E3           580 RNCHKLP5:
83E3 ED 5B 33  581          LD      DE,(CLRWK2)     ;KOUSIN SHINAI
83E6 88
83E7 C9        582          RET
83E8           583 ;
83E8           584 ;
83E8           585 CLEAR:                           ;KESU
83E8 11 B9 86  586          LD      DE,CLRWK
83EB 1A        587          LD      A,(DE)          ;NAINARA RETURN
83EC B7        588          OR      A               ;END CODE=00H
83ED 28 51     589          JR      Z,RETURN
83EF 3E 2A     590          LD      A,'*'           ;HAJIME HA '*' DE
83F1 06 02     591          LD      B,2             ;       TSUGI HA ' '
83F3           592 CLRLP:
83F3 08        593          EX      AF,AF'
83F4 1A        594          LD      A,(DE)
83F5 B7        595          OR      A
83F6 28 38     596          JR      Z,CLRLP2
83F8 6F        597          LD      L,A
83F9 13        598          INC     DE
83FA 1A        599          LD      A,(DE)
83FB 67        600          LD      H,A
83FC 13        601          INC     DE
83FD CD 1E 20  602          CALL    #LOC
8400 08        603          EX      AF,AF'
8401 CD F4 1F  604          CALL    #PRINT
8404 08        605          EX      AF,AF'
8405 D5        606          PUSH    DE
8406 11 35 88  607          LD      DE,FALLWK       ;KESHITA RETU NI ATARU
8409 7D        608          LD      A,L             ;       FALLWK NO
840A D6 10     609          SUB     10H             ;       FLAG WO TATERU
840C 83        610          ADD     A,E
840D 5F        611          LD      E,A
840E 3E 00     612          LD      A,0
8410 8A        613          ADC     A,D
8411 57        614          LD      D,A
8412 3E 01     615          LD      A,1             ;FLAG=1
8414 12        616          LD      (DE),A
8415           617          ;
8415 E5        618          PUSH    HL              ;SCORE KASAN
8416 3A 27 85  619          LD      A,(HEIGHT)
8419 21 33 85  620          LD      HL,POINTS
841C 86        621          ADD     A,(HL)
841D 27        622          DAA
841E 77        623          LD      (HL),A          ;6 BITE WO
841F 3E 00     624          LD      A,0             ;       2 BITE ZUTSU
8421 23        625          INC     HL              ;       10 SHIN HOSEI
8422 8E        626          ADC     A,(HL)
8423 27        627          DAA
8424 77        628          LD      (HL),A
8425 3E 00     629          LD      A,0
8427 23        630          INC     HL
8428 8E        631          ADC     A,(HL)
8429 27        632          DAA
842A 77        633          LD      (HL),A
842B E1        634          POP     HL
842C D1        635          POP     DE
842D 08        636          EX      AF,AF'
842E 18 C3     637          JR      CLRLP
8430           638 CLRLP2:
8430 21 00 50  639          LD      HL,5000H        ;KARA LOOP
8433           640 CLRLP3:
8433 2B        641          DEC     HL
8434 7D        642          LD      A,L
8435 B4        643          OR      H
8436 20 FB     644          JR      NZ,CLRLP3
8438 11 B9 86  645          LD      DE,CLRWK
843B 3E 20     646          LD      A,' '           ;KESU CHARACTER
843D 10 B4     647          DJNZ    CLRLP
843F           648 CLRLP4:
843F C9        649          RET
8440           650          ;
8440           651 RETURN:
8440 E1        652          POP     HL
8441 C3 C7 80  653          JP      MAIN
8444           654 ;
8444           655 ;
8444           656 ALFALL:                          ;KESHITA ATONO RAKKA
8444 11 3D 85  657          LD      DE,CHKWK
8447 ED 53 B7  658          LD      (CHKWK2),DE
844A 86
844B 06 09     659          LD      B,9             ;TATENI 9 RETSU
844D 2E 10     660          LD      L,10H
844F 11 35 88  661          LD      DE,FALLWK
8452           662 RCHECK:                          ;RETSU GOTONI CHECK
8452 1A        663          LD      A,(DE)          ;CHECK SUBEKI RETU?
8453 B7        664          OR      A
8454 C4 6D 84  665          CALL    NZ,FLCHECK      ;FLAG GA TATTEIRU?
8457 2C        666          INC     L
8458 13        667          INC     DE
8459 10 F7     668          DJNZ    RCHECK
845B ED 5B B7  669          LD      DE,(CHKWK2)     ;END CODE=00H
845E 86
845F AF        670          XOR     A               ;       WO DAINYUU
8460 12        671          LD      (DE),A
8461 3A 3E 88  672          LD      A,(BELLWK)
8464 3D        673          DEC     A
8465 C0        674          RET     NZ
8466 32 3E 88  675          LD      (BELLWK),A      ;UE KARA OCHITA TOKI
8469 CD C4 1F  676          CALL    #BELL           ;       DAKE BELL
846C C9        677          RET
846D           678 FLCHECK:
846D AF        679          XOR     A               ;FLAG WO TAOSU
846E 12        680          LD      (DE),A
846F 26 16     681          LD      H,22            ;Y=22
8471           682 SKCHECK:                         ;SHITA KARA CHECK
8471 E5        683          PUSH    HL
8472 D9        684          EXX
8473 E1        685          POP     HL
8474 06 00     686          LD      B,0             ;B=SHITAKARA IKUTSU ME
8476           687 SKCHECK2:                        ;       MADE BLOCK GA
8476 CD 1B 20  688          CALL    #SCRN           ;       TSUZUITE
8479 FE 20     689          CP      20H             ;       IRUKA?
847B 28 04     690          JR      Z,UPCHECK
847D 04        691          INC     B
847E 25        692          DEC     H
847F 18 F5     693          JR      SKCHECK2
8481           694          ;
8481           695 UPCHECK:
8481 26 16     696          LD      H,22
8483 AF        697          XOR     A
8484 B0        698          OR      B               ;ICHIBAN SHITA GA
8485 28 03     699          JR      Z,RFALL         ;       BLOCK NASHI
8487           700 UPCHKLP:
8487 25        701          DEC     H               ;BLOCK GA NAKUNARUMADE
8488 10 FD     702          DJNZ    UPCHKLP         ;       Y GENSHOU
848A           703 ;
848A           704 RFALL:
848A CD 1B 20  705          CALL    #SCRN
848D FE 20     706          CP      20H
848F C4 9A 84  707          CALL    NZ,RFALL2       ;BLOCK ARINARA RAKKA
8492 25        708          DEC     H
8493 7C        709          LD      A,H             ;ICHIBAN UE?
8494 FE 01     710          CP      01H             ;       Y=1?
8496 20 F2     711          JR      NZ,RFALL
8498 D9        712          EXX
8499 C9        713          RET
849A           714 RFALL2:                          ;HITOTSU ZUTSU OTOSU
849A E5        715          PUSH    HL              ;Y HOZON
849B F5        716          PUSH    AF              ;CHECK CHARA HOZON
849C           717 RFALLLP:
849C 24        718          INC     H
849D CD 1B 20  719          CALL    #SCRN
84A0 FE 20     720          CP      20H             ;MADA OTOSERU?
84A2 20 12     721          JR      NZ,RFALL3
84A4 25        722          DEC     H               ;HITOTSU MODOSHITE
84A5 CD 1E 20  723          CALL    #LOC            ;       SPACE WO
84A8 CD F1 1F  724          CALL    #PRINTS         ;               HYOUJI
84AB 24        725          INC     H               ;HITOTSU FUYASHITE
84AC CD 1E 20  726          CALL    #LOC            ;       BLOCK WO
84AF F1        727          POP     AF              ;               HYOUJI
84B0 F5        728          PUSH    AF
84B1 CD F4 1F  729          CALL    #PRINT
84B4 18 E6     730          JR      RFALLLP
84B6           731 RFALL3:
84B6 ED 5B B7  732          LD      DE,(CHKWK2)     ;POINTER WO POP SHITE
84B9 86
84BA 25        733          DEC     H               ;       CHKWK NI
84BB 7D        734          LD      A,L             ;       DAINYUU
84BC 12        735          LD      (DE),A
84BD 13        736          INC     DE
84BE 7C        737          LD      A,H
84BF 12        738          LD      (DE),A
84C0 13        739          INC     DE
84C1 ED 53 B7  740          LD      (CHKWK2),DE     ;POINTER KOUSHIN
84C4 86
84C5 21 3E 88  741          LD      HL,BELLWK       ;UEKARA OCHITARA
84C8 36 01     742          LD      (HL),1          ;       BELL WO NARASU
84CA F1        743          POP     AF
84CB E1        744          POP     HL
84CC C9        745          RET
84CD           746 ;
84CD           747 ;
84CD           748 GMOVR:                           ;GAME OVER NO SHORI
84CD CD C4 1F  749          CALL    #BELL
84D0 CD C4 1F  750          CALL    #BELL
84D3 21 10 0B  751          LD      HL,0B10H
84D6 CD 1E 20  752          CALL    #LOC
84D9 CD E2 1F  753          CALL    #MPRINT
84DC 20 20 20  754          DM      "         "
84DF 20 20 20
84E2 20 20 20
84E5 00        755          DB      00H
84E6 24        756          INC     H
84E7 CD 1E 20  757          CALL    #LOC
84EA CD E2 1F  758          CALL    #MPRINT
84ED 20 52 45  759          DM      " RETRY ? "
84F0 54 52 59
84F3 20 3F 20
84F6 00        760          DB      00H
84F7 24        761          INC     H
84F8 CD 1E 20  762          CALL    #LOC
84FB CD E2 1F  763          CALL    #MPRINT
84FE 20 20 20  764          DM      "         "
8501 20 20 20
8504 20 20 20
8507 00        765          DB      00H
8508           766 GMOVRLP:
8508 CD D0 1F  767          CALL    #GETKY
850B B7        768          OR      A
850C 28 FA     769          JR      Z,GMOVRLP
850E FE 59     770          CP      'Y'
8510 CA 00 80  771          JP      Z,8000H
8513 FE 4E     772          CP      'N'
8515 28 02     773          JR      Z,END
8517 18 EF     774          JR      GMOVRLP
8519           775 END:
8519 3E 0C     776          LD      A,0CH
851B CD F4 1F  777          CALL    #PRINT
851E C9        778          RET                     ;RETURN TO S-OS
779 ;
780 ;
781 ;        WORK & DATA
782 ;
783          ;
784 BLKXY:                          ;SANREN BLOCK NO ZAHYOU
785          DS      2
786 NOWBLK:                         ;IMANO BLOCK NO CHARACTER
787          DS      3
788 NXTBLK:                         ;TSUGINO BLOCK NO CHARACTER
789          DS      3
```

▶X1turbo版グラディウスには感動してしまった。あんな素晴らしいものを作れる人が，同じユーザーとしてうらやましく思います。私も何年かかってもいいから，自分のやりたいことをやってみたいと心底思います。　　直樹　広明(23)茨城県

```
790 HEIGHT:                     ;BLOCK WO OTOSHITA TAKASA
791         DS      1
792 CHRTBL:                     ;BLOCK NO CHARACTER TABLE
793         DB      'A','O','X','M','K','S'
794 ;
795 OLDRND:                     ;MAENO RANSUU NO DATA
796         DW      0E933H
797 STEP:                       ;RANSUU HASSEI DE SHIYOU
798         DW      899
799 REFR:                       ;R REGISTER NO BUFFER
800         DS      1
801 ;
802 POINTS:                     ;SCORE
803         DS      3
804 BLKS:                       ;BLOCK NO KAZU
805         DS      2
806 ;
807 KEYBF:                      ;INKEY ROOTINE DE SIYOU
808         DS      1
809 ;
810 SPEED:                      ;GENZAINO SPEED
811         DS      2
812 SPDWK:                      ;SPEED NO COUNTER
813         DS      2
814 ;
815 CHKWK:                      ;KIERUKA DOUKA NO CHECK WORK
816         DS      9*21*2
817 CHKWK2:                     ;CHECK WORK NO POINTER
818         DS      2
819 CLRWK:                      ;KESERU BLOCK NO ZAHYOU DATA
820         DS      9*21*2
821 CLRWK2:                     ;CLEAR WORK NO POINTER
822         DS      2
823 ;
824 FALLWK:                     ;RAKKA CHECK NO X ZAHYOU
825         DS      9
826 ;
827 BELLWK:                     ;BELL WO NARASU (KESITA ATONO
828         DS      1                                RAKKA)
829 ;
```

全機種共通システムインデックス

■85年6月号
序論　共通化の試み
第1部　S-OS"MACE"
第2部　Lisp-85インタプリタ
第3部　チェックサムプログラム
■85年7月号
第4部　マシン語プログラム開発入門
第5部　エディタアセンブラZEDA
第6部　デバッグツールZAID
■85年8月号
第7部　ゲーム開発パッケージBEMS
第8部　ソースジェネレータZING
■85年9月号
インタラプト　S-OS番外地
第9部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表
■86年1月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年2月号
第15部　S-OS"SWORD"
第16部　Prolog-85入門(1)
■86年3月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年4月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年5月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年6月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ&エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶ magiFORTH
特別付録　PC-8801版S-OS"SWORD"
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版 S-OS"SWORD"
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版 S-OS"SWORD"
■86年9月号
第28部　FuzzyBASIC 発表
連載　明日に向かって magiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタ DREAM
第31部　FuzzyBASIC 料理法〈1〉
■86年11月号
第32部　パズルゲーム HOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC 料理法〈2〉
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC 料理法〈3〉
■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC 料理法〈4〉
■87年2月号
第36部　アドベンチャーゲーム MARMALADE
第37部　テキアベ作成ツール CONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツール MAGE
付録　"SWORD" 再掲載と MAGIC の標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS"SWORD" 変身セット
第43部　MZ-700用 "SWORD" を QD 対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASIC コンパイラ
第45部　エディタアセンブラ ZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージ JACKWRITE
特別付録　FM-7/77版 S-OS"SWORD"
■87年9月号
第49部　リロケータブル逆アセンブラ Inside-R
特別付録　PC-8001/8801 版 S-OS"SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASIC コンパイラの拡張
第52部　X1turbo 版 S-OS"SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OS の仲間たち
第53部　もうひとつの FuzzyBASIC 入門
第54部　ファイルアロケータ&ローダ
インタラプト　S-OS こちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo 版 "SWORD" アフターケア
　　ラインプリントルーチン
特別付録　PASOPIA7 版 S-OS"SWORD"
■88年1月号
第58部　FuzzyBASIC コンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲーム ELFES
■88年3月号
第60部　構造型コンパイラ言語 SLANG
■88年4月号
第61部　デバッギングツール TRADE
第62部　シミュレーションウォーゲーム WALRUS
■88年5月号
第63部　シューティングゲーム ELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語 SLANG 入門(1)
第66部　Lisp-85 用 NAMPA シミュレーション
■88年7月号
第67部　マルチウィンドウドライバ MW-1
連載　構造化言語 SLANG 入門(2)
■88年8月号
第68部　マルチウィンドウエディタ WINER
■88年9月号
第69部　超小型エディタ TED-750
第70部　アフターケア WINER の拡張
■88年10月号
第71部　SLANG 用ファイル入出力ライブラリ
第72部　シューティングゲーム MANKAI
■88年11月号
第73部　シューティングゲーム ELFES Ⅳ
■88年12月号
第74部　ソースジェネレータ SOURCERY
■89年1月号
第75部　パズルゲーム LAST ONE
第76部　ブロックゲーム FLICK
■89年2月号
第77部　高速エディタアセンブラ REDA
特別付録　X1版 S-OS"SWORD"〈再掲載〉
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOROBAN
■89年4月号
第79部　SLANG 用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータ RING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲーム　TICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲーム PUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリ
　　DIO. LIB
■90年1月号
第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ
■90年2月号
第89部　超小型コンパイラTTC++
■90年3月号
第90部　超多機能アセンブラOHM-Z80
■90年4月号
第91部　ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部　インタプリタ言語STACK
■90年6月号
第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲーム　SQUASH！
第95部　X68000対応S-OS "SWORD"
特別付録　PC-286対応S-OS "SWORD"
■90年7月号
第96部　リロケータブルアセンブラWZD
■90年8月号
第97部　リンカWLK
■90年9月号
第98部　BILLIARDS
■90年10月号
第99部　ライブラリアンWLB
■90年11月号
第100部　タブコード対応エディタEDC-T
■90年12月号
第101部　STACKコンパイラ

＊以上のアプリケーションは，基本システムである S-OS "MACE" または S-OS"SWORD" がないと動作しませんので ご注意ください。

THE SENTINEL

▶先日，引っ越しをしたらOh!X (Oh! MZ)だけでダンボール1箱になりました。今度の引っ越しのときには手伝ってください。　藤田　伸夫(24)新潟県

愛読者プレゼント(Part2)

さてさて，先月に引き続き3周年記念プレゼントです。今月はソフトハウス編。
けっこうな量のプレゼントでしょ？　なんと今月だけで245名に当たるのだから，競争率は低くなるってもんだね。さあどれにするかが決まったらさっそくハガキを出してちょーだいな。
応募方法は168ページを見てね。

1

日本ファルコム　☎0425(27)6501

A.ドラゴンスレイヤーカードケース　10名
B.イースIIIカードケース　10名
C.リリアカードケース　10名
D.リリアスタンプ　10名
E.リリアバッジ　5名
F.リリアキーホルダー　5名
G.卓上カレンダー　30名
H.ファルコムバッジ　5名
I.ファルコムキーホルダー　5名
J.JDKバッジ　5名
K.JDKキーホルダー　5名

お馴染み日本ファルコムからは，こぉ～んなにたくさんのプレゼントグッズをいただきました。これだけで，なんと100名分！　ファルコムファンは見逃せませんね。

2

ツァイト　☎03(299)0461

A. 3D倶楽部リビング
X68000用　5"2HD版　9,700円(税別)　1名
B. 3D倶楽部ダイニング
X68000用　5"2HD版　9,700円(税別)　1名
C. ねじ式
X68000用　5"2HD版4枚組　12,800円(税別)　1名

ツァイトからは，CGツールのデジタルクラフトのデータライブラリの3D倶楽部，そしてあのねじ式をプレゼント。各1名の方に。

3

東芝EMI　☎03(587)9005

A. ビデオ ヴァリスIII　3,500円(税込)　5名
B. CD FZ戦記 アクシス　2,500円(税込)　3名
C. CD AIRCOMBAT/IMPERIALFORCE/大戦略III'90　2,500円(税込)　3名
D. CD グラナダ　2,500円(税込)　3名
E. CD ラグーン　2,500円(税込)　3名

あの優子がなんとアニメーションになりました。このぶんだと，X68000にヴァリスIIIが登場するのもそう遠いことではないかな。CDのほうは最新のゲームミュージックをプレゼントです。

4

イマジニア　☎03(343)8911

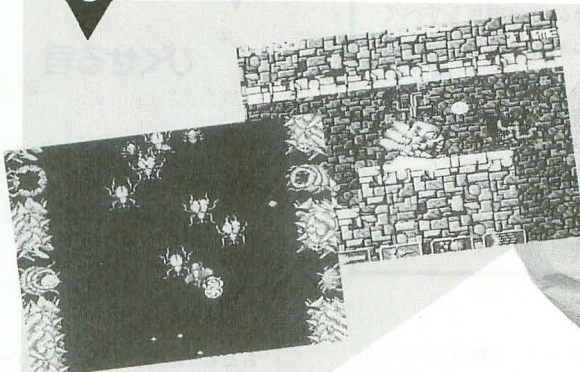
A.シムアースプロモーションビデオ　10名
B.シムアースポスター　10名
C.ポピュラスポスター　10名

はやくもイマジニアから発売が決定したシムアース，ゲームのほうはもうちょっと先だけど，ビデオとポスターをプレゼントしちゃいます。ポピュラスファンの人のためにこちらのポスターも用意しました。

5 ポニーキャニオン ☎03(221)3161

A. ウルティマTシャツ 4名
B. ウルティマステッカー 30名
C. ウルティマシャープ&ボールペン 6名

ポニーキャニオンからは，あの有名なウルティマシリーズ，ウルティマⅣのグッズをプレゼント。どれもほかではちょっと手に入らない代物です。

6 システムサコム ☎03(635)7609

A. ジェミニウイングオリジナルブルゾン 1名
B. ジェミニウイング X68000用 5″2HD版2枚組 8,800円(税別) 1名
C. アトミック・ロボキッド X68000用 5″2HD版 8,800円(税別) 1名

ジェミニウイングの発売記念に作ったオリジナルブルゾンを1名の方に。そしてアーケードから移植されたバリバリのシューティング2種類を各1名に。

7 BNN ☎03(238)1323

A. X68000 X-BASIC入門 2,600円(税込) 5名
B. X68000 3Dグラフィックス入門 2,300円(税込) 5名
C. OS-9/X68000操作ガイド 2,300円(税込) 5名

X68000をゲームマシンとしてしか使ってない人ってわりといるはず。でも，ここらで自作のゲームを，って思っている人にはこれらの本は重宝するぞ。さあ，これであなたもプログラマ！

8 マイクロプローズジャパン ☎0423(33)7781

A. GUNSHIP
X68000用 5″2HD版 14,800円(税別) 2名
B. GUNSHIP アパッチプラモデル 2名

本格フライトシミュレータとあって，人気の高いGUNSHIP。このゲームは攻撃用ヘリコプター「アパッチ」を操縦するってなわけで，今回はその「アパッチ」のプラモデルもあるのです。

9 T&E SOFT ☎052(773)7770

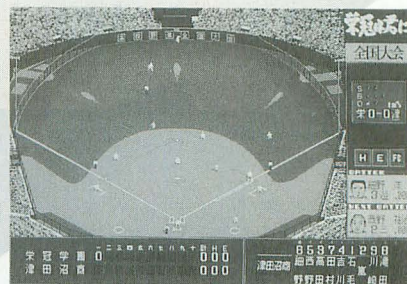
幻獣鬼

X68000用 5″2HD版3枚組
8,800円(税別) 2名

「遥かなるオーガスタ」ばかりが取りざたされているT&Eだけど，このゲームもなかなかのもの。転職システムや，豊富なアイテムなどでプレイヤーを飽きさせないアクションゲームです。

10 アートディンク ☎0474(77)7541

栄冠は君に

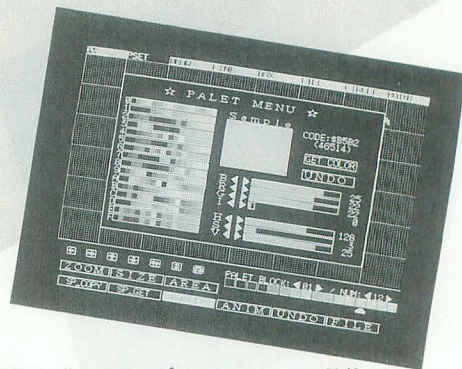
X68000用
5″2HD版3枚組
9,500円(税別)
3名

高校野球をシミュレートしたゲーム。バントや牽制がやたらあるのもらしくていい。約4,000校の中から1校を選び，目指すはもちろん甲子園，そして全国大会優勝だ。深紅の旗が待っているぞ。

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください(同番号内にいくつか種類がある場合は2-A，2-B……のようにアルファベットも明記のこと)。締め切りは1991年1月18日の到着分まで。当選者の発表は1991年3月号で行います。

M.N.M Software　☎0423(60)3084

ぴくせる君

X68000用　5″2HD版
4,800円(税込)　3名

もともとはM.N.Mの自社ゲーム開発用に作られたスプライトエディタ。機能はそんなに多くはないけれど，そのぶん低価格で使いやすいツールだ。

12　日本ソフテック　☎0425(82)1502

LUCY・SHOT

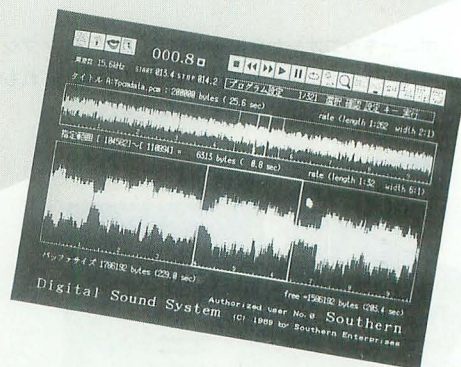
X68000用　5″2HD版2枚組　7,800円(税別)
3名

ソフテックのピンボールシリーズ第2弾。今回は前作の「PINBALL・PINBALL」とうってかわってアメリカン・コミック調。ちなみに背景の美少女の名前はルーシーです。

13　サザンエンタープライズ　☎03(787)3932

DISS-P

X68000用　5″2HD版
7,800円(税込)　2名

録音・再生・編集が手軽にできる多機能デジタルサウンドツール。録音再成はもちろん，編集もマウスでらくらくというスグレモノ。X68000にしゃべらせるスピーチ機能もついているぞ。

14　光栄　☎045(561)6861

オリジナルトランプ

10名

信長の野望シリーズ最新作　信長の野望　武将風雲録」のイラストが印刷されたトランプ。おじさんがこっちをにらんでいるのがなんともコワイ。これで神経衰弱だけはしたくないと思う今日この頃。

ザイン・ソフト　☎078(242)2855

オリジナルフロッピーケース

10名

コンピュータを扱っていて，わりと不便だなと思うのがフロッピーの持ち運び。で，このかっこいいザインのフロッピーケースの登場だ。これを友達に自慢してやろう。くやしがるぞ。

11月号プレゼント当選者

1サイバリオン（福島県）斎藤繁（大阪府）森秀樹（鳥取県）渡辺直之　2幻獣鬼（埼玉県）北大路晃聡（神奈川県）藤田勝幸（奈良県）梶田真二　3PINBALL・PINBALL（宮城県）米田龍生（愛知県）山元伸一（兵庫県）今田智宣　4ニュージーランドストーリー（北海道）池上匠（埼玉県）松崎剛史(鹿児島県）福留敏和　5サイバリオンCD　（東京都）田代俊一（新潟県）滝沢政憲（石川県）山下智史
以上の方々が当選されました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，公正取引委員会の告示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

X68000 マシン語プログラミング 入門編

著●村田敏幸

B5判変型・388ページ　定価2800円(税込み)

ストロングスタイルのマシン語入門!!

●本書まえがきから

1990年11月現在。『Oh！X』誌に連載中の「X68000マシン語プログラミング」のうち＜入門編＞と称した冒頭部分を1冊にまとめたら，こんな本になった。マシン語プログラミングに興味をもったX68000ユーザーのための副読本，とでもいったらよいのだろうか。少なくとも，教科書的なプログラミング入門書では決してない。むしろ，問題集であり，実践テキストのノリに近い。

マシン語にかぎらず，プログラミングに関する知識/技術は，実際のプログラミングの中でこそ身につき，磨かれるものだ。この不変の真理にもとづき，本書は読者に自分の頭と体とを使うことを強いるように書かれている。エッセンスを100倍くらいに薄めて吸い差しでとろとろと流し込むような親切さは排除した。文書の襞に隠れた大小の謎は，サンプルプログラムを読み，動かし，改良することによって解き明かされるだろう。

●本書の内容

'91年1月刊行予定

SX-WINDOW プログラミング

吉沢正敏●著

B5判変型　予価3200円(税込み)

X68000にイベントドリブン方式のマルチタスク・ウィンドウ環境を提供するSX-WINDOWは，X68000に新たな世界を拓くものとして熱い期待を集めている。本書は，このSX-WINDOW上でプログラムを作ってみたいと思っているユーザーを対象にした，プログラム作成のためのガイドブックである。イベントドリブン，リソースなどのウィンドウ・プログラムの基礎知識，サンプルプログラムによる具体例，ウィンドウ関連のシステムコール一覧など，SX-WINDOW上でプログラミングする際のエッセンスを集めている。

BACK ISSUES

バックナンバー案内

ここには1990年1月号から12月号までをご紹介しました。現在1989年10～12，1990年2，6，8～12月号までの在庫がございます。バックナンバーおよび定期講読のお申し込み方法については，188ページを参照してください。

1990

1月号（品切れ）
特集1　オペレーティングスタイルの研究
特集2　Cプログラミング応用編
連載　ショートプロぱーてぃ/Z80's Bar
X68000マシン語/C調言語講座/DōGA・CGA
●X1/turbo用シミュレーションゲームSuper Battle
LIVE in '90　さよならを過ぎて/RYDEEN
THE SOFTOUCH　レナム/メタルサイト
全機種共通システム　WORM KUN/再掲載SLANG
特別付録　X68000 THE SOFTWARE CATALOGUE

2月号
特集　画像圧縮へのアプローチ
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
X68000マシン語/C調言語講座/X-BASIC調理実習
●X68000用ゲームプログラムGon Gon
●MZ-700用紙芝居Eyelarth
LIVE in '90　オーダイン/魔女の宅急便
THE SOFTOUCH　A-JAX/フラッピー2/夢幻戦士ヴァリスII
マジックパレット/Mu-1/CYBERNOTE PRO-68K
全機種共通システム　超小型コンパイラTTC++

3月号（品切れ）
特集　MUSICアドベンチャー
X68000用MIDIドライバ&音源エディタ
なんでも鳴らせるOPMD.X/MMLを楽譜データに
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
C調言語講座/X-BASIC調理実習
●X1/turboシミュレーションCRISIS in Tokyo
LIVE in '90　パワードリフト/スキーム/となりのトトロ
THE SOFTOUCH　ナイトアームズ/斬/ダンジョンマスター
全機種共通システム　超多機能アセンブラOHM-Z80

4月号（品切れ）
特集　ゲームシステム文学誌
1989年度GAME OF THE YEAR発表
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
X-BASIC調理実習/C調言語講座/X68000マシン語
●X1・MZ-2000/2500用RPG The Cave of Dalk
●うわさの68040，ついに登場
LIVE in '90　バーニングフォース(OPMD対応)
THE SOFTOUCH　The Fille Professor/HOST PRO-68K
全機種共通システム　ファジィコンピュータシミュレータI-MY

5月号（品切れ）
特集　BASICプログラミング
第5回　言わせてくれなくちゃだワ
連載　ショートプロぱーてぃ/Z80's Bar
X-BASIC調理実習/X68000マシン語プログラミング
●新機種X68000SUPER-HD/EXPERTII/PROII
●ラジコンスティックの製作
LIVE in '90　TURBO OUTRUN
THE SOFTOUCH　天下統一/ポピュラス/Hyperword
全機種共通システム　インタプリタ言語STACK

6月号
特集　創刊8周年記念PRO-68K(付録5"2HD)
Oh!Xアンケート結果大分析大会
連載　ショートプロぱーてぃ/Z80's Bar/PurePASCAL
X-BASIC調理実習/X68000マシン語プログラミング
●X1turbo用コマンドシェルシミュレータ
●ハードウェア工作入門
LIVE in '90　ナイトアームズ/悪魔城伝説/この木なんの木
THE SOFTOUCH　三国志II/FAR SIDE MOON/グラナダ
全機種共通システム　X68000用S-OS"SWORD"他

7月号（品切れ）
特集　マシン語への第一歩
X68000SUPER-HD試用レポート
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
X-BASIC調理実習/PurePASCAL
●INTEGRAL XI――ノーマルXIへの対応
●ハードウェア工作入門
LIVE in '90　夢幻戦士ヴァリスII/トッカータとフーガニ短調
THE SOFTOUCH　サーク/あーくしゅ/ダウンタウン熱血物語
全機種共通システム　リロケータブルアセンブラWZD

8月号
特集　ADVANCED 2D GRAPHICS
100号記念特別モニタプレゼント
連載　ショートプロぱーてぃ/Z80's Bar/INTEGRAL XI
X-BASIC調理実習/X68000マシン語プログラミング
PurePASCAL/ハードウェア工作入門
●X68000用画像回転プログラム XROT0.X
LIVE in '90　OMENS OF LOVE/ENDLESS RAIN/ダートフォックス
THE SOFTOUCH　大航海時代/ウルティマV/プロミストランド
全機種共通システム　リンカWLK

9月号
特集1　日本語を処理するための序章
特集2　ADVANCED 2D GRAPHICS
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
X-BASIC調理実習/マシン語プログラミング
PurePASCAL/ハードウェア工作入門
●清水和人流プログラミング道場
LIVE in '90　風の谷のナウシカ/ラジオ体操第一
THE SOFTOUCH　T&T/D-Again/シムシティー/ギャラガ'88ほか
全機種共通システム　BILLIARDS

10月号
特集　電子音楽術入門
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
マシン語プログラミング/ハードウェア工作入門
清水和人流プログラミング道場
●荻窪圭の大人のためのX68000
●中森章のようこそここへC言語
LIVE in '90　Rise And Fall/PARADOX/キューピー3分クッキング
THE SOFTOUCH　ワールドコート/ルーンワース/闇の血族/提督の決断
全機種共通システム　ライブラリアンWLB

11月号
特集　理科系のGAME REVIEW
連載　Z80's Bar/DōGA・CGA/カードゲーム
マシン語プログラミング/ハードウェア工作入門
PurePASCAL/X-BASIC調理実習
ようこそここへC言語/INTEGRAL XI
荻窪圭の大人のためのX68000
LIVE in '90　ピラミッドソーサリアン/ザ・スキーム
THE SOFTOUCH SPECIAL　ラグーン/幻獣鬼/サイバリオン/GUNSHIP他
全機種共通システム　スクリーンエディタEDC-T

12月号
特集　XCのための傾向と対策
連載　X-BASICプログラミング調理実習/ハードウェア工作入門
マシン語プログラミング/ショートプロぱーてぃ/Z80's Bar
大人のためのX68000/ようこそここへC言語/INTEGRAL XI
●シミュレーションプログラミング入門
●特別企画アナログジョイスティックの製作
LIVE in '90　グラディウスIII/メタルサイト
THE SOFTOUCH SPECIAL　イメージファイト/ジェミニウイング/NAIOUS他
全機種共通システム　STACKコンパイラ

［第8話］

パソコン戦線異状なし

TAKAHARA HIDEKI 高原 秀己

珍しくパソコンの話。

ぼくの周りを見ると，最近になって，結構楽しくパソコンを仕事場で使いはじめた人が何人かいる。

そう，ブック型とかノート型という，薄型ラップトップ機を使っているのだ。

これまではこわごわワープロを清書機代わりに使っていただけの人が，

「パソコンはやっぱりOSを使えなきゃあ，ね」

としたり顔をして解説しながら，一太郎などを使って悦に入っているのを見ると，バカバカしいのを通り越して微笑ましくもある。

彼らはいずれも今年になって，PC-9801n(98ノート）とかPC-286note Fを買ったクチである。

さて笑ってないで，自分のことを考えてみると。

とんでもないことに気がついてしまった。

ぼくはこの1年，パソコン(X68000とかじゃなく,ごくごくありふれた機械)をなにひとつパワーアップしていないのである。例外的に「エコロジーII」と「ノートン・ユーティリティ」を買ってユーティリティを活用することになったのを除けば，ソフトもハードも含めてまったく変化のない1年であった。

もちろん，その他のエレクトロニクス製品にしたところで，スーパーファミコンを買ったわけでもなく，BSもレーザーディスクもまたまた買わずじまいだった。

だが，スーパーファミコンは注文する意欲がなかっただけだし，BSとレーザーディスクはレンタルビデオですませた，という大義名分があるからいい。いずれにしても，なければないでいいものだったから，さして問題ではない。

ところがパソコンを1年間何もシステムアップしなかったのは，まだパソコンが黎明期を脱したにすぎない時代であることを考えると，実に問題だ。

もしも，技術革新が激しかったのに，ぼくが何も適応しようとしなかったとすれば，これは衰退を意味する。革新への追随を止めるということは，研究所が新しい研究をせず，工場が新たな設備投資を止め，そしてスーパーや外食企業が新たな店舗を開設しなかったことと同じなのだ。

ぼくは新しい技術に目を向けなかったのだろうか？

そのことを確認するために，パソコン関連ビジネスの今年の出来事を列記してみると……。

・ノート機商戦が本格スタート。
・ビジネス用でi386高速版，i486マシンが続々登場。
・LAN管理ソフトをめぐりメーカー/ソフトハウスで派閥分かれる。
・ハードウェアがますます大容量化。
・PCエンジンのカードソフトがそのまま使えるポータブル機が発表，この後，ベールを脱いだスーパーファミコン。
・4MビットダイナミックRAMが量産開始に。

こんなとこだろうか。

このうちゲーム機はジャンルが異なるのでオミット。4Mはまだパソコン向けに普及する段階ではないのでパス，ハードディスクの低価格大容量化はこれまでの延長なのでパス，ましてやLAN管理ソフトは新会社ができただけなので無視，とはじいていくと，実に冒頭の光景として紹介したノート型パソコンの普及といった程度しか残らない。この他で記憶に残っていることといえば，エプソンがリンドバーグをCMに使ったり，PC電子手帳が出たり，と実にくだらないことばかり。

実際のとこ，革新的なことは何もなかったようだ。WINDOWSは雑誌のオマケにはなったが使わなくてはならない状態にはいたっていないし，日本語環境を変える新しいフロントエンドプロセッサが登場したわけでもない。画期的なプリンタが登場したわけでもなく，メモリカードも普及しなかった。

ソフトにしても，「ナントカ2」や「アイカワラズ3」といったパート2ものばかりで，画期的なブームを呼んだゲームが出たわけでもなく，革新的なビジネスツールにもお目にかからずじまい。

そう！　結論として，ぼくが悪かったわけではなく，マーケットにあまりにも変化がなかったのだ。

改めて考えてみると，今年ぼくは「なにか新しくて役に立つものはないかな」と何度秋葉原に行ったことか。よく行く新宿のパソコンショップでビックリするようなゲームやプリンタを探したが，一度として見つかったことはなかったのだ。

かくいうぼくの使い方は，パソコン通信用の端末がほとんどで，あとはワープロ，各種分析用の1-2-3（表計算ソフト)，ゲームといった，きわめて固定的なシンプルなものだけに，「なにか」というのがなかなか難しいことはわかる。

それでも革新的なものさえあれば，50万円規模の投資は考えていただけに，ちょっと残念だ。

そもそも，ノート型パソコンの普及のみが目立ったということは，PC-9801ならびにエプソン製互換機がまたまた躍進したことを示しているだけだ。NECを追う富士通だのシャープだのが巻き返しをした気配はカラキシない。

あまりに何もなさすぎたのだが，このままの状態が続くようであれば，来年もヒマつぶしに大幅値引きになっているノート機を買って終わり，という状態になるのがオチだろう。

だが本当はそんなことではいけないはずだと思う。まだまだパソコンは常に新しいシーンをぼくたちに見せ続けてくれるべき時期を脱していないはずだ。

変に安定成長を決め込んで，成長を拒否するようでは先が見えてしまう。刺激のないマーケットは退廃を生むだけ。退廃すれば，それで終わりだ。

1991年に期待。

ジョブズはやっぱり天才だ！

スーパースターの来日

たしかにこう感じました。「スティーブ＝ジョブズはやっぱり天才だ！」。いまもまだ輝きの残るAppleII，毎日お世話になっているMacintosh，そして新機種発表で普及時期に突入したNeXT。この魅力あふれる3台のマシンを世に送り出してくれたジョブズが来日し，1時間半にわたる圧倒的なプレゼンテーションを見せてくれました。10月1日から5日まで開かれていた情報処理学会30周年記念国際会議“InfoJapan”の最終日の目玉として彼の招待講演が催されたのです。

会場である京王プラザホテルのホール入り口前では開場前から多くの人が待っていて，これから始まろうとするジョブズの講演に対する期待が充満していました。なるべく前に座りたいと思い，早くから待っていたところ（前から4列めぐらいにもぐりこめた），すっと横をあの人なつっこい顔が通りすぎていきました。ずいぶんと背が高いな，びっくりすると同時にそう感じました。ふてぶてしいまでの余裕にあふれた態度は相変わらずといったところでしょうか。

ジョブズは講演前日の夕方，40人（あるいはfourteenかも）のスタッフとともに来日したそうです。そして，スタッフたちは徹夜で会場設備のセッティング，ジョブズ本人も朝8時からリハーサルを行ったという話も聞かれ，期待も高まる一方といったところです。

さらにうれしいことには，発表されたばかりの新しいNeXTも持ってきていて，デモンストレーションが行われたのです。今回の講演は情報処理学会という格式高い学会の主催であり，タイトルも「’90年代のコンピューティング」ということで，新しいNeXTの宣伝が前面に出ることはないにしても，やはり，新しいNeXT，そう，「カラーネクスト」に対する期待はいやがおうにも高まります。

圧倒的なプレゼンテーション

ジョブズは長身ながらステージの上を軽く動き回ります。そして，時々観客に問いを投げかけます。「どうですか？」という軽いものから，「これこれがわかる人いますか？」という質問まで。話はパーソナルコンピュータの発展史における革命を順番にたどりながら進んでいきます。ジョブズらしく，ソフトウェアにおける革命のみを対象としています。

まず第1の革命がスプレッドシートです。それを担ったマシンとしてIBM-PCが引き合いに出されます。そして，会場の向かって右側のほうにセッティングされていた新しいNeXTにインストールされている“Improv”というソフトの紹介（画面は正面の特大スクリーンに投影されます）。あまりピンとはきませんでしたが，この“Improv”は新しいNeXTのウリのひとつらしいのです。ジョブズの持つ「1990年代のコンピューティング」の姿を少しずつ明らかにしながら，実質的には新しいNeXTの宣伝をやっているようなものでした。

第2の革命はデスクトップパブリッシングです。これを担ったマシンとして，Macintoshが挙げれます。この説明では特にNeXTを使ったデモンストレーションはありませんでしたので，ちょっと気になりました。疑問に思った人が質問したのに対しては別に意味はないと答えていました。新しいNeXTではワープロ関係では特に目新しいバージョンアップがなかったのでしょう。

第3の革命はカスタムソフトです。はっきりとした説明はなかったのですが，ソフトウェア作成環境などをさしているようです。マシンとしてはSUNワークステーションが示されます。ここでは，NeXTのインタフェイス作成支援ソフトウェアであるインタフェイスビルダのデモンストレーションがありました。最近，NeXTをちょっといじっているので特に驚きませんでしたが，少しバージョンアップしているようです。

いよいよ第4の革命。これこそがNeXTが背負って立つ，今後5年間のキーワードということです。それは，“Interpersonal Computing”です。個人個人を結びつけるグループ全体を考えたコンピューティングが極めて重要であるということを，ジョブズは強く主張します。考えてみれば，1980年代のパーソナルコンピューティングは文字どおり個人個人の知的環境を拡張するものでした。そして，それが拡大すればするほど，ほかの人の領域との交わりも広がってきます。そのための，自然かつ綿密な融合こそが“Interpersonal Computing”の目的だということです。別の見方をすれば，計算機はメディアとしての第1歩を踏み出していくということなのでしょう。

新しいNeXTを使って，この第4の革命の一端が示されます。拡張された電子メール機能の紹介が主でした。単なるテキストから始まり，ボイスメール，チェロ奏者ヨーヨーマの演奏，計算機プログラム。それから，これも今度のNeXTのウリらしいのですが，FAXも自由自在に転送してみせました。内蔵のOCRソフトが，FAXイメージのテキスト領域をASCIIテキストに変換してくれるそうです（モデムはオプション）。

圧巻のカラー実演

このようにして，講演は4つの革命を追って淡々と進んでいきます。4つめの革命“Interpersonal Computing”を担うのはNeXTなのだと強くアピールして，一応ひとつの筋は終わります。しかし，ここから観衆の待ち望んでいたことが始まりました。いうまでもなく，新しいNeXTのカラー機能の実演です。

新しく用意されたカラー機能にもいろいろバリエーション（選択肢）がありますが，真のカラー（True Color）を実現するには32BITカラーボードを接続する必要があります。このボードにはインテルのi860（33MHz）とJPEG形式のデータ圧縮プロセッサが搭載されています。これにより，リアルタイムにビデオを取り込み圧縮したり，解凍して表示することができます（1.4Gバイトのハードディスクを接続すればなんと1時間分といわれています）。

ジョブズはステージ右のほうに設置された新しいNeXTを説明しながら操作し，観客は正面の大スクリーンを見つめます。なにかの映画の一部がNeXTの上で再現されます。カラーの絵を加工するソフトはなかなかのものでした。スポーツカーの窓ガラスごしに見える景色がリアルに見えるときには思わず観客席からため息が洩れていました（マウスでスポーツカーをつかんで高速に動かしたりもしました）。

フィナーレは息を飲むほどあざやかでし

た。カメラでジョブズ自身の上半身をリアルタイムに取り込みます。手のひらを上に向けたところを取り込んだかと思うと，ドナルドダックの絵を切り取ってきて，それをその手のひらの上に乗せたのでした。どっと拍手が湧きました。500人ほどの観衆たちはことばさえ出せずに，ただただ拍手をしつづけていました。

アメリカ人の質問

最後は自由に質問を受けつけ，それにジョブズが答えるというものでした。残念ながら，日本人はほとんど質問をせずアメリカ人ばかりでしたが，そのうちのいくつかを簡単に紹介しましょう。

＊　＊　＊

Q．UNIXで標準的になりつつあるウィンドウシステム，X-WINDOWについて。
A．X-WINDOWはMacintosh以下である。NeXTをベストなUNIXマシンと見る人がいるが，別にUNIXマシンを作るのが目的ではない。1％にすぎないUNIXユーザーのためのマシンを作ったのではないのだから。

＊　＊　＊

Q．CISC対RISCについてどう思うか?
A．速いRISCができたのならば，それはすばらしいと思うし，もっといいCISCができたのならば，それはよかったと思う。どんなに速いプロセッサができたかということより，速いプロセッサを使ってどのようなマシンを作るかということにこそ，意味があるのだから。

＊　＊　＊

ジョブズのこの2つの答えは，誰もが使えるマシンにするということ（ヒューマンインタフェイス指向），どういうことをマシンにやらせるかということ（ソフトウェア指向）をストレートに表しており，Macintosh, AppleIIなどを生む思想の本質的な部分に関わる考え方であるということができると思われます。ところで，ジョブズに対してある人がこう質問しました。

＊　＊　＊

Q．Virtual Realityについてあなたはどう思いますか？

＊　＊　＊

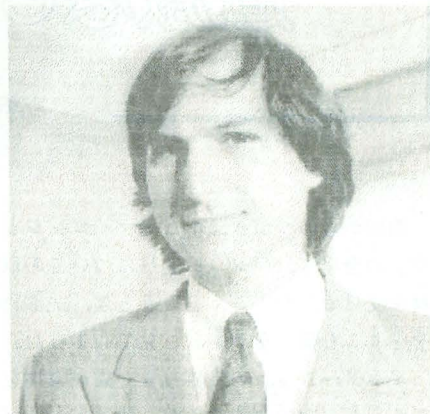
それに対してジョブズは具体的なことはなにも言いませんでした。でも，大きな興味を持っているというようなことだけは言っていました。

仮想現実とは

Virtual Reality，日本語に直せば「仮想現実」です。最近，この概念はちょっとしたブームになっているようで，特集を組んでいる雑誌もあります。この概念は，現実ではないのに現実のように感じさせる技術のことを表しています。というとすぐに，脳に直接信号を入れるようなSF的な話を連想してしまいますが，そこまでいくような話はいまのところないようです。

人間を入出力をもつ装置と考えるとき，理論的には，入力のみ，つまり五感に対して正確な情報を与えれば，それが実在しない仮想的なものでも人は現実だと思い込んでしまいます。たとえば，コンタクトレンズ状のもので正確に映像が映るものを目につけ，鼻と耳と舌にも気がつかないくらい小型でしかも高性能な匂いと音の発生器と味覚刺激器をつけておきます。それで，それらを統一した場面を構成するように刺激を同時に与えて再現してやれば，人もだまされるだろうというわけです。

でも，触覚は難しいものがあるでしょう。人が出力をも同時にもった装置であることから発生する別の難しい問題と絡むからです。たとえば，手でものをつかむというごく当たり前の出力（行動）を考えてみましょう。腕の回りにセンサでもつけて腕の行動を検知します。腕を動かすと目につけたコンタクトレンズ型表示器のなかの腕も動かしてやります。ここで，ほかの物体をつかむような状況を仮想的に実現させようとするときには難しいことになります。表示器につかもうとした物体をうまく表示するだけでなくて，手のひらの適当な位置にものがあたったという感触を与えてやらなければならないからです。この制御をリアルタイムでやるのは大変な問題でしょう。

この仮想現実はいまのところごく限られたことにしか応用されていないようですが，ヒューマンインタフェイスとしては究極的なものであることには間違いがないでしょう。とんでもない応用が無限に考えられるような気がしませんか？

とびっきりの美女との関係

最近，NeXTを自由に使うことができるようになって，暇を見つけてはいじってます(といっても古いNeXTですが)。気に入った点，素晴らしい点はいくらでも書けますが，いやな点はひとつだけです。それは，有名なことかもしれませんが，やたらと遅いことです。もともと，とても静かなマシンなので，待たされているときでもハードディスクを読んでいるからか，プロセッサがなにか計算でもしているからか，わからなくて，暴走したのかなと一瞬不安になることもよくあるほどです。

しかし，今度のNeXTはCPUが68030から68040に変更されたということもあり，この一大欠点はかなり解消されたのではないでしょうか。

ジョブズによると，古いNeXTのもつ，1）値段が高い，2）速度が遅い，3）ソフトが少ない，4）カラーが出ない，という問題点を新機種はすべて解決したとのことです。4）はたしかでしょう。3）はどうだかはっきりはわかりません。そして，1）と4）の問題は同時には解決していないような気もしますが，総合的には見事なマシンを発表してくれたなという気がします。

ジョブズの顔に絶えず浮かんでいる笑みとあふれるような自信。生まれつき，あるいは，噂によるととびっきりの美女と結婚したばかりとのことで，そのことも手伝っているのかもしれません。しかし，僕にはAppleII, Macintosh, と生んできた彼がいよいよ，真打ちを生み出したと実感している表情に思えてなりませんでした。

第55回

猫とコンピュータ

過激なCRTと共に

Takazawa Kyoko
高沢 恭子

前回に引き続き「第7回ホビーマイコンショウ」の模様をお届けします。これを書き上げるのにも並々ならぬ苦労があったようで……。でも，猫の話題にはついつい目がいくところがキョウコさんらしいですよね。

前回からのつづき,「ホビーマイコンショウ」の会場で，パソコン通信についての取材を受ける話を書こうというとき,「洗濯物をとりこむ猫」をテレビで見てしまった。

アニメーションの世界なら，猫が家事をやろうが，政治をやろうが，夢は広がるばかりかもしれないが，これは家庭用ビデオで撮影した実写だったので，なんだか困ったような気持ちになった。

白黒模様のその猫は，共稼ぎの中年のご夫婦に飼われていて，無人の留守宅で洗濯物をとりこんでいたそうだ。はじめは誰がやったのかわからず不気味だったが，あるとき現場を見て知った。人に言っても信じてもらえなかったので，苦心してビデオにおさめたらしい。

マンションのベランダで，猫はハンガーの洗濯物をひとつずつ口でひっぱってははずしては，室内に運びこんでいた。少し不鮮明な画面がいかにもドキュメンタリーらしくて，映像の説得力はやはりすごいなと思った。

少し困ったなと感じたのは，人のいないところでは，イタズラか悪いことをやる賢さのあるのが猫だったのに，かくれた善行らしいことをやっている猫がいて，証拠ビデオつきで公表されてしまったことだ。

でも，人間とかけひきをするのがじょうずな猫が，ほめてくれる人もいないところで,「お手伝い」なんてムダなことをするだろうか。これは猫の哲学に反する。

とっても可愛がられていたこの猫は，たぶん主人夫妻の帰りを待つ毎日をくりかえして大きくなった。夜になると，帰宅した奥さんはまず洗濯物をとりこみ，それからこの猫にゴハンを与える。猫の頭の中では，洗濯物のとりこみとゴハンは，ワンセットの連鎖的な行事になった。

夕方からだんだん夜になり，寂しいしおなかもすいてくる。ふと奥さんの帰宅したときのようすが猫の頭に思い浮かぶ。そうだ，洗濯物だ，ベランダの洗濯物をとりいれると，つぎはゴハンだ。猫は洗濯物を室内に入れることで，連鎖的に奥さんの帰りと晩ゴハンを実現できるような期待を持ったのだろう。

「拾われて可愛がられたことへの恩返しだと思う」というご主人の言葉にあえてさからって，猫はじぶんのためだけに行動すると，主張したい。

火を吹くCRT

と，ここから前回のつづきを書こうとして，キーをひとつ叩いたとたん，突然CRTは真っ暗になり，バチバチというおそろしい音をあげはじめた。

ブラウン管の中で火花が暴れつづけ，いまにも破裂する気配も感じられたので，これからどうなるのかを見届けたい気持ちを押さえて電源をきった。

もちろんテキスト画面をセーブするゆとりもなくスイッチをきったので，ここまでの文章は思いだしながらの，2度目の入力になった。セーブのアクションをしていたら，パソコン本体は損傷していたかもしれない。

CRTを本体から分離して，もう一度電源を入れてみた。こんどは「ピーッ」という大きな鋭い音をあげて，緑色の火花が背面部であざやかに光った。

テレビが古くなったり，ホコリがたまって発火した話は聞いたことがあるけれど，このCRTはまだ4年とちょっとしか使っていない。室内全体もそれぞれのマシンの周囲も，できるだけ清潔に保つことを心がけてきたつもりなのに，こんなマンガのような現実がわが家でおこるなんて。

驚いたりあきれたりしながら，つくづくCRTをながめた。ともかくメインになっていちばん働いてくれたCRTだったが，そんなにひどい扱いをしていたのかな。ほとんど1日つけっぱなしという日もあるけれど，OA機器はそれがふつうだ。いったい中身はどんなようすなのか，ついでに背中のフタをはずしてのぞいてみた。

ホコリはとくに目立たない。患部確認のために電源をつないでみる。いちばんうしろの部分の細い線の束から，激しい火花と音が出た。ただのショートかもしれないが，いよいよ爆発寸前の様相になってきたので現場保存にとどめた。夫はいつも，帰宅のたびにコワイ事件が待っているので，疲れるだろうな。

本体をX68000のCRTにつないで，うまいぐあいにエディタのしごとは継続できた。

通信鬼ケ島

やっと，前回からのお話。

FBI-NETのメンバー，編集マンの銀猫さんからの紹介で，某女性週刊誌の記者氏とカメラマン氏があらわれた。場所はアキバのラジオ会館8階大ホール,「第7回ホビーマイコンショウ」開催中の会場。

取材の目的は，女性ネットワーカーたちそれぞれの，パソコン通信とのかかわりかたについてらしい。取材される面々は，FBIの女性メンバー6人。うち3人は20代のお嬢さんたちで，この1年くらいのあいだに加入した新メンバーだ。

パソコン通信の効用が，このところいちだんと言われるようになってきた。すでに大きな役割をはたしている商用データベースは別格として，企業内で，大学や研究所で，お役所，公共の施設，サービス機関な

どで，つぎつぎに新しいネットワークの手段として採用されはじめている。

情報の共有，選択の任意性，保管，転送などの点で，同じパーソナルな通信でも，電話やファックスにはない活用法を，たくさん持っているといえる。

取材に訪れた週刊誌を発行している大手S社は，パソコン誌も出していて，社内ではいくつもの雑誌がそれぞれに通信ネットを持っているそうだ。この日の「女性S」誌も「Sネット」という通信局をおいて，読者との交流をはかっているが，ほんの少数がアクセスしているだけとのこと。パソコン通信の話題が高まってきたこの機会に，もう1歩踏み込んで，理解と普及につとめてみようというつもりらしい。

まずはひとりずつの写真撮影。ショウの展示に使っていたマシンの前で，6人が順番にキーボードを叩くしぐさをしてみせる。パソコンの取材では，このワンパターンのポーズはどうしても覚悟しなければならない。カメラマン氏がしごとを終えて先に帰ると，ホールの奥にある控室で，取材となった。

「銀猫」さんからは，取材記者氏がパソコンとは無縁の方ということを聞いていたが，それはとても良い点がある半面，お互いになかなかの苦労があるだろうと予想できた。「パソコン通信」は，パソコンを使って通信をするだけのことであっても，マシンの面と，ネットワークとしての面の双方から，成立ち，しくみ，特徴や特異性までもふくめて，全容を説明するのはたいへんなことだ。ふと，2年くらい前に，いまよりもっとパソ通が未開のころ，趣味の話をする座談会で「通信」からの代表で出席して，とてもむずかしかった記憶がよみがえった。

「銀猫」さんも同席して，若い記者O氏の質問がはじまる。パソコン通信をはじめた動機，チャンス，あるいは目的，そしてじっさいの利益や楽しみはなんなのかといったことが，端から順に問われていくのだが，用語の1つひとつに注釈が必要なので，なかなか進行が悪い。

ホスト局，ホストプログラム，端末プログラム，シスオペ，アクセス，ID，パスワード，メニュー，ボード，階層構造，アーティクル，オンライン，アップロード，ダウンロード……などなど，そして通信族の特殊用語や表記術，OFF会の存在や交際の流儀までと，キリがない。

それを説明しているのが，「ちやこ」さんに，「えいりあん」さっちゃん，「ちゃーりー」さん，「exit.」さん，「鳩」さんといった，お互い本名もよく知らないハンドルネームの一団だ。真剣に，いぶかしそうに質問をつづけるO記者が，鬼ヶ島を訪れた桃太郎みたいに見えてきた。

パブリックをめざして

先日NHKテレビの「婦人百科」で「絵本づくり」を指導しているのを見て，なんとなくパソコン通信を連想した。

手づくり絵本のしごとは，オリジナルな絵とお話をつくることが半分で，もう半分はその手で製本することだった。わが子や孫へ心をこめた贈りものをしたいという動機から，愛好者も多いらしい。

もう何冊か手づくりの絵本を製作した人たちの話では，もちろん語り聞かせたい絵とお話があっての「絵本」ではあるけれど，ハードの分野ともいえる製本にかける喜びはとても大きいようだ。このあたりが現状でのパソコン通信にとてもよく似ていると感じた。

パソコン通信も，目的はネットワークづくりとコミュニケーションであるはずだ。でもいまはコミュニケーションの内容より，パソコンという手段の新しさや有能さが注目を浴びている。

キーボードから叩き出した情報が，モデムの仲立ちで電話回線に乗せられ，遠く離れたパソコンに送られるのだ。もともと，情報の移動や，編集，整理分類，収納が得意なパソコンが，距離と空間を越えてその力を発揮できるようになったのだ。これは注目されるだけのことはある。

だからうっかりすると，こんなふうに話題だけで取り上げて，「さあ通信をはじめましょう」と呼びかける一方で，ますます高度で有能な「パソコン通信」が開発されて，じっさいには手軽さから遠ざかっていってしまう心配もある。

パソコン通信は，男女を問わず，なんの用意もない人が取りかかろうとしたら，そんなに簡単にはいかない。意欲も知識も根気もいる。費用もかかる。いちばんたいせつな「目的」を持ちつづけられなければ，ムダに終わる。

女性週刊誌だから女性を取材するのはしぜんなことだけれど，独力ではじめる女の人は少ないかもしれない。ここにいるメンバーも，じつは好都合な環境に負うところが大きい。「ちやこ」さんは新聞社のワープロオペレータ，「えいりあん」さっちゃんはシスオペ夫人，「ちゃーりー」さんはプログラマー，「exit.」さんは東電につとめる電気のプロ，「鳩」さんは通信機器メーカーの社員だ。

みんなじょうずに質問に答えて，パソコン通信の楽しさを語っている。良い記事に仕上がって，読者の共感を呼ぶものになればいいと思う半面，結果的に不親切な報道をまたひとつつくることにならなければいいなと感じたひとときだった。

* * *

ところで，キーを叩いている椅子のそばに，背中が黒くコゲついたCRT，PC-KD851が死んでいる。やっぱり新しいCRTをひとつ買わなくてはダメかしら。X68000といちいちつなぎかえていたのでは，能率もカッコも悪いものネ。

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

SCSI内蔵，チタンブラック

X68000 SUPER

シャープ

X 68000 SUPER

シャープは，「X68000 SUPER-HD」のハードディスクなしのモデル，「X68000 SUPER」(CZ-604C-TN)，および，チタンブラックの専用ディスプレイ「CZ-606D-TN」を発売した。

「X68000 SUPER」は「X68000 SUPER-HD」と同じようにSCSIインタフェイスを標準装備しているので，先頃発売された光磁気ディスクなどのSCSI規格の周辺機器を接続できる。

また，標準ではハードディスクは内蔵していないが，追って発売される増設用3.5インチハードディスク「CZ-68H」(80Mバイト，アクセスタイム19ms)が内蔵可能である。

価格は「CZ-604C-TN」が348,000円，「CZ-606D-TN」が79,800円(ともに税別)，「CZ-68H」(平成3年2月発売予定)は価格未定。

〈問い合わせ先〉

シャープ㈱　☎03(260)1161，06(621)1221

「書院」シリーズ，新4機種

WD-A321/341/630/730

シャープ

シャープは，毛筆体も内蔵した液晶表示ラップトップモデル「WD-A324/341」と，「トリプルフォント&マルチアウトラインフォント」搭載のCRTモデル「WD-A630/730」の合計4機種を発売した。

「WD-A324/341」は「書院スーパーアウトラインフォント」に新たに毛筆体も内蔵し，名刺に使えるような小さな文字から，垂れ幕やポスターに使えるような大きな文字まで104種類の文字サイズをアウトラインフォントで印刷できる。価格は178,000円と198,000円（ともに税別）。

「WD-A630/730」は明朝体，ゴシック体（ROM)，毛筆体（正楷書体；FD）のトリプルフォントを標準で装備，さらに，毛筆体（行書体）や教科書体の別売フォントを加えたマルチフォントの拡大文字（4倍角以上の文字）すべてに対し，アウトラインフォント機能が自動的に働くようになっている。価格は145,000円と178,000円（ともに税別）。

〈問い合わせ先〉

シャープ㈱　☎03(260)1161，06(621)1221

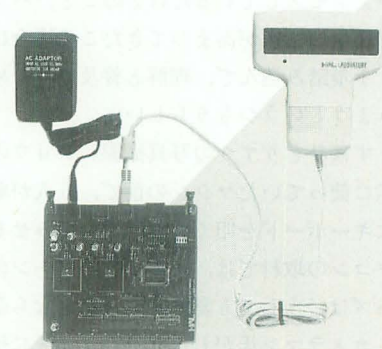
WD-A341

WD-A730

ハンディイメージスキャナ

ファインスキャナ-X68

HAL研究所

ファインスキャナ-X68

HAL研究所はX68000専用のハンディイメージスキャナ「ファインスキャナ-X68」を発売する。

「ハンディスキャナ-X68」はハンディスキャナでは最高クラスのグレイスケール256階調取り込みを実現している。読み取り幅は105mm。専用インタフェイスボードによる高速転送で20mm/sの移動速度での読み取りが可能。

専用ソフト「IMAGE PHOTO 68」では取り込み画像をメモリ上に持ったまま画面を見ながら微調整することが可能で，従来の明るさやコントラストなどを変えながら何度も取り込みを行う手間を解消している。

また，調整項目のγ補正ほか細かい項目がユーザー指定できる。特にユーザー曲線による輝度調整は強力。

画面出力モードに関わらず，メモリ上に256階調データを持っているので，拡大/縮小操作を行っても画面の美しさは損なわれない。このため，画面の解像度とプリンタの印字密度の違いを十分に吸収できる。

そのほかZIMをはじめ多彩なファイル形式をサポートしている。

価格は未定だが本体，インタフェイスボード，IMAGE PHOTO 68込みで4万円

程度に抑えられる模様。モノクロながら高品位な画面が得られるツールとして期待される。
〈問い合わせ先〉
㈱HAL研究所　☎03(252)5561

アナログRGBをS端子に
XAV-1S
電波新聞社

電波新聞社はアナログRGB信号の入力をビデオあるいはS端子信号に変換して出力できるアダプタを12月末に発売する。

入力は21ピンRGBマルチ，15ピンアナログ（15KHz，200ライン，標準解像度）に，出力はS端子，ビデオ信号に対応していて，外部ACアダプタによる電源供給が必要。

パソコンなどのアナログRGB出力はもちろん，キャプテンシステムターミナルの出力や文字放送のRGB出力をS端子に変換するなどの用途も考えられる。価格未定。
〈問い合わせ先〉
電波新聞社　☎03(445)6111

XAV-1S

TVゲーム用バトルシート
MENKURI
アイアンクラフト

アイアンクラフトは，ジョイスティックを固定することのできるTVゲーム用シート「MENKURI」を12月21日から発売する。

この「MENKURI」は市販されているジョイスティックを固定してプレイすることにより，操作性，操作フィーリングを向上させようというもの。ジョイスティックホルダー部分は前後に80mmスライドするので使用者の体型や好みに応じてセッティングできる。また，脚部にはプラスチック製のアジャスタがついていて，安定したプレイが可能。

対応する機種は，以下のとおり。
アスキースティックXターボ（アスキー）
サイバースティック（シャープ）
XE-1シリーズ(APは不可)(電波新聞社)

価格は7,800円（税込）＋1,000円（送料）で，パソコン誌などを媒体とする通信販売のみで取り扱われる。
〈問い合わせ先〉
アイアンクラフト　☎0256(33)6111

MENKURI
INFORMATION

第5回 ファンタジー&ロールプレイングゲームフェア

大阪の駸々堂書店では毎年恒例の「ファンタジー&ロールプレイングゲームフェア」を'90年12月1日から'91年1月15日までの期間，開催する。

○商品：ファンタジーシミュレーション，テーブルトークRPG，カードゲーム，メタルフィギュア，ダイス，専門誌バックナンバー，関連図書，コミック，ゲームブック以上の販売，および，テーブルトーク，カードゲーム全商品1割引き，「ファンタジーRPGイラスト」の募集，展示などの特別企画が予定されている。
〈問い合わせ先〉
駸々堂書店　☎06(353)4011

イマジニア協賛による
杉本彩コンサート

ポピュラス，シムシティなどのヒットで今年はおおいにパソコンゲーム界を賑わせてくれたイマジニア。そのイマジニアがファンサービスの一環として，杉本彩のコンサートの協賛をする。
○タイトル　みんなゴージャス！
○場所　渋谷クラブ・クワトロ

○日時　1月7日（月）　開演　19:00

イマジニアでは，このコンサートにペア75組，150名を招待，応募方法は官製ハガキに，
1)　名前
2)　住所
3)　電話番号
4)　雑誌名（Oh!X）
を明記したうえ，
〒163　東京都新宿区西新宿2-7-1新宿第一生命ビル13階　イマジニア株式会社
杉本彩コンサート係
まで。平成2年12月25日必着。
〈問い合わせ先〉
イマジニア㈱　☎03(343)8911

BOOK

OS-9/68000 マルチユーザーシステムガイド

本書は「RAINBOW OS-9ガイド」に続くOS-9ガイドブック第2弾であり，OS-9/68000の最大の機能であるTSS(Time Sharing System)によるマルチユーザー機能を解説したものである。X68000版完全対応。価格は3,300円(税込)。
〈問い合わせ先〉
スピリットパブリッシング　☎0258(92)4818

FILES Oh!X

このインデックスは，タイトル，注記——筆者名，誌名，月号，ページで構成されています。いよいよ1991年に突入，あけましておめでとう。さて受験生の人，あともう少しです，コタツで眠っちゃダメですよ。

参考文献
I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
テクノポリス　徳間書店
ポケコンジャーナル　工学社
POPCOM　小学館
マイコン　電波新聞社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

一般

▶ネットワーカー・ホリック　第30回
「ぼのぼの」でおなじみの漫画家，いがらしみきお氏作「パソ通疑似体験ゲームBBSちゃん」を紹介。PDSはPC-9801用ホストプログラム「mmm」「TURBO-BBS」。——編集部，LOGIN，21号，272-273pp.

▶ネットワーカー・ホリック　第31回
オンラインゲーム特集。ネットワークを使って，他のアクセス者と楽しめるネットワークマルチプレイヤーゲームを紹介。——編集部，LOGIN，22号，272-273pp.

▶レッツ・トライ・コンピュータミュージック
主にPC-9801についてコンピュータミュージックの始め方や製品紹介を行っているが，X68000も含めたソフトのデータ互換一覧表やMIDIの規格についての記事も掲載されている。——藤本健・山田憲一，マイコン，12月号，121-154pp.

▶キーボードのなくなる日
最近注目度が高まってきているGUIの現状について，Macintosh，MS-WINDOWS，NeXTなどを対象に見ていく。——編集部，マイコン，12月号，172-177pp.

▶MUNEPI♪のゲーム探検隊
AMショーリポートの巻。各社大型きょう体の新作ゲームが目立ったほか，ガンシューテイングのゲームやテーブルゲームにも魅力ある作品が出品されていたとのこと。——MUNEPI♪，マイコン，12月号，236-237pp.

▶やまさんのアルゴリズム・ブック
最終回。リスト処理の一環として，バイナリツリー構造の考え方を解説しプログラムを作る。——やまさん，マイコン，12月号，254-258pp.

▶ビジネスマンの情報活用術
シャープから発売されたハイパー電子システム手帳PA-9500について特徴と機能，周辺機器などさまざまな面から紹介。——塚田洋一，マイコン，12月号，326-328pp.

▶実践ハード入門
ガイガーカウンターの原理と製作。今回は部品入手などの事情から秋月電子のキットを組み立てる。放射線と放射能に関する解説記事つき。——石川至知，マイコン，12月号，378-381pp.

▶第2回サイクロンCG大会
9月24日に行われた第2回サイクロンCG大会の模様と受賞作品の紹介をしている。——編集部，ASCII，12月号，440p.

▶シャープハイパー電子システム手帳 DB-Z
シャープの新型電子手帳PA-9500は，タッチパネルの採用，高精細画面，手書き入力など今までの電子手帳にない高機能を搭載している。その機能と，これからの期待について述べる。——塚田洋一，ポケコンジャーナル，12月号，4-10pp.

▶プログラムBASICカードエミュレータ BE
電子手帳のプログラムBASICカードのプログラムを支援するソフトウエア“BE”の構成とクロス開発の方法についての解説。——塚田洋一，ポケコンジャーナル，12月号，19-24pp.

▶なんでもQ&A
All in Note用のモデムについて，パソコン通信を行うには，自動起動の解除方法は，などの質問に答える。——編集部，マイコン，12月号，426-427pp.

MZシリーズ

MZ-1500(BASICMZ-5Z001)

▶ミューラア
キミは魔道師ミューラアを倒せるか？　敵弾をよけつつ，敵本体に体当たりするという，アクションゲーム。——FROG，マイコンBASIC Magazine，12月号，124-126pp.

MZ-2500(BASIC-M25)

▶ヘンテコヘンテコMAZE
見知らぬ塔の中の迷路を通って，つぎの階の入り口へいく。全7面のへんてこりんパズルゲーム。——坂井重典，マイコンBASIC Magazine，12月号，127-128pp.

X1/turbo/Z

X1シリーズ

▶TECLA
落ちてくるカラーブロック。色を合わせてぶっ壊せ！——久保開，マイコンBASIC Magazine，12月号，154-155pp.

▶LAST MISSION
ズオを8方向に移動させ，鉄球を振り回して敵にダメージを与える。しかし面ごとに違った命令があり，それに従わなくてはならないのがミソ。——佐井川師治，マイコンBASIC Magazine，12月号，156-158pp.

X1+FM音源ボード（要FM音源ドライバ）

▶MSX版パロディウス ～エンディング～
ゲームミュージックプログラム。——桐畑厚宏，マイコンBASIC Magazine，12月号，188-189pp.

X1turboシリーズ

▶NEW SOFT
新着ゲーム，プール・オブ・レイディアンスを紹介。——編集部，LOGIN，21号，27p.

▶Cyber Mission
横3重スクロールシューティング。——伊藤敬之，マイコンBASIC Magazine，12月号，159-161pp.

新刊書案内

年末が近づくと，そこかしこの出版社から翌年の技術動向とかトレンディとかキーワードがどうとかいう本が出版される。今回はそんな本のなかから，電波新聞社のエレクトロニクスキーワード集を選んでみた。1項目につきキーワード＋関連項目という構成が非常に読みやすかったからだ。真面目に読む科学技術の本ではなく，暇潰しにぺらぺら眺めるのが一番正しい読み方。今の世の中，一寸先はファジーと呼ばれるほど，わけのわからないカタカナ用語が溢れかえっている。年末くらい，こういった軽い読み物でカタカナをチェックしてみるのもいいだろう。収録項目が少ないのも読み物向きだ。Macintoshの関連事項にEMSが入っているという不思議な面や「ワープロの進化はパソコンにとってはひとつの脅威である」といった間抜けな記述（今はパソコンの小型低価格化がワープロにとって脅威なのだ）もあるが，おおむね，楽しんで1990年に注目された技術や古くからあってもまだ注目する技術がわかる。

巻末にある付録の各種データもなかなか興味深くて面白い。(K)

エレクトロニクスキーワード集　1991年度版電波新聞社編　電波新聞社刊　☎03(445)6111　A5版233ページ　1,900円

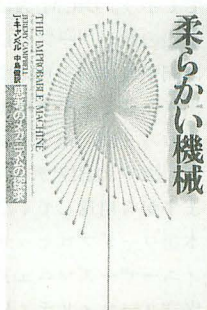
X68000

▶NEW SOFT
アトミック・ロボキッド，バルーサの復讐，ダイナマイト・デュークほか，グラフィックツールのCANVAS PRO-68Kを紹介。——編集部，LOGIN，21号，16-31pp.
▶最新ゲーム徹底解剖!!
ラグーン，ナイアス，サイバリオン，ジェミニウイングを徹底攻略。——編集部，LOGIN，21号，144-175pp.
▶X68000新聞
ニューラル・ギア，シュヴァルツシルト，イメージファイト，C compiler PRO-68K ver2.0のその後。新作情報はダイナマイト・デューク，銀河英雄伝説II，Magical Shot。——編集部，LOGIN，21号，252-255pp.
▶NEW SOFT
ソル・フィース，生中継68の紹介。——編集部，LOGIN，22号，25-26pp.
▶X68000新聞
シャープ秋の新製品「XBAStoC CHECKER」「SCSIボード」「光磁気ディスクユニット＆ディスクカートリッジ」を紹介。新着ゲームは「続ダンジョン・マスター」「ダイナマイト・デューク」「銀河英雄伝説II」，日本語入力フロントエンドプロセッサ「FIXER Ver.4.0」。——編集部，LOGIN，22号，252-255pp.
▶最新ゲーム徹底解剖!!
アクションRPG「ラグーン」を攻略。——編集部，LOGIN，22号，150-153pp.
▶NAGDRV FOR X68K
発表！ FM，AD PCM，MIDIに対応したミュージックドライバ。コマンドリストとプログラム本体を掲載。——永田英哉＆あんど，マイコンBASIC Magazine，12月号，69-82pp.
▶Spell of Vision III
画面中央の自分を黄色の壁で跳ね返らせて，ゴールを目指す。——菅野憲昭，マイコンBASIC Magazine，12月号，162-163pp.
▶剣道
うまく間合いを取り，勝ち抜いていき，敵の大将を倒す。ジョイスティック専用。——林純一，マイコンBASIC Magazine，12月号，164-165pp.
▶ROD LAND ～1話エンディングBGM～
ジャレコのゲームミュージックプログラム。——M.H，マイコンBASIC Magazine，12月号，190-191pp.
▶バーニングフォース ～Grass Land～
同号のマイコンBASIC Magazineで発表されたNAGDRV用のミュージックデータ。バーニングフォースのBGM。要MT32系MIDI楽器。——永田英哉，マイコンBASIC Magazine，12月号，192-194pp.
▶MS-DOS入門ブック
MS-DOSの入門用解説書だが，コマンドリファレンスなどHuman68kを使うX68000ユーザーにも参考になる。——編集部，マイコンBASIC Magazine，12月号，別冊付録
▶チャレンジ!! アドベンチャー・ゲーム
闇の血族・上巻を，画面写真で紹介。——佐久間亮介，マイコンBASIC Magazine，12月号，225-227pp.
▶SOFT RADAR
続ダンジョン・マスター，ピンボール・ピンボールを紹介。——編集部，POPCOM，12月号，10-11，21pp.
▶ゲームがオレを呼んでいる！
シムシティーのレビューとラグーンの攻略法，ナイアスの紹介など。——編集部・たかはぴ・アビゲイル金万，POPCOM，12月号，70-85pp.
▶WE ARE THE X68000 WORLD
アトミック・ロボキッド，闇の血族・完結編，ダイナマイト・デューク，パロディウスだ！，生中継68，ニューラル・ギアなどの新着，開発中ゲームと，光磁気ディスク，SCSIインタフェースボードを紹介。——編集部，POPCOM，12月号，90-93pp.
▶ミュージック・パビリオン
「パラダイスの確率（JAJA）」のミュージックプログラム。——編集部，POPCOM，12月号，167-170pp.
▶GAMING WORLD
ナイアス，アクシス～FZ戦記～，Magical Shot，バルーサの復讐，闇の血族・完結編，ニューラル・ギア，ダイナマイト・デュークを紹介。——編集部，テクノポリス，12月号，18-33pp.
▶SOFT EXPRESS
続ダンジョン・マスター，ピンボール・ピンボールを紹介。——編集部，コンプティーク，12月号，91，97p.
▶X68000 SPIRITS
ラグーンの攻略法，遥かなるオーガスタ，シュヴァルツシルト，ダイナマイト・デュークを紹介。——編集部，コンプティーク，12月号，252-255pp.
▶ゲームレビュー
熱血高校ドッジボール部サッカー編とナイアス，アクシス～FZ戦記～の批評記事。——IWAKICHI・あゆさわかつみ・相川春利，マイコン，12月号，225-235pp.
▶なんでもQ&A
付属ワープロで数種類の文字列を格納しておく方法，TYPEコマンドでプリンタのティアオフ機能を使うには，ビデオレセプターとX68000の相性について，など。——編集部，マイコン，12月号，424-425pp.
▶AVプログラミング講座
複数のスプライトを表示・移動する場合のテクニックをスプライト機能を基に説明する。——宮本親一郎，ASCII，12月号，361-368pp.（リスト481-492pp.）
▶AV STRASSE
X68000用SCSIボードと光磁気ディスクユニットの製品概要，CANVAS PRO-68Kの批評記事が掲載。——宮本親一郎，ASCII，12月号，409-412pp.
▶FREE SOFTWARE INDEX
ここ数ヵ月の間に主要ネットにアップロードされたPDSの一覧。X68000用のフロッピー高速化プログラム・アニメーションツールなどの情報も収録されている。——編集部，ASCII，12月号，450-454pp.
▶長期ロードテスト
シリーズ第4回。X68000の音楽環境について。PDSなどの充実と，BGMを鳴らしながらほかの作業ができるなどの機能に感心している。——編集部，ASCII，12月号，468-470pp.
▶GAME BOX
闇の血族，ラグーン，熱血高校ドッジボール部サッカー編の紹介・批評記事。——吉沢正敏・市原昌文・牛島健雄，I/O，12月号，130-135pp.
▶PIX.C
以前発表された画像ローダのバージョンアップ版。アセンブリ言語で記述してスピードアップされ，オプション機能も充実。——Zenobia，I/O，12月号，186-187pp.
▶XBAS to C CHECKER
シャープから発売されたXBAS to C CHECKERについて，変換エラーの具体例や動作テストを通じてその性能を探る。——市原昌文，I/O，12月号，198-201pp.

ポケコン

PC-E500
▶とってもDIFFICULT
オセロに似たパズルゲーム。とっても難しいぞぉ。——有我裕明，マイコンBASIC Magazine，12月号，170p.
▶Varunas
防衛システムのコンピュータが暴走，我が基地が標的にされた！ バリバリのシューティングゲーム。——山本拓矢，マイコンBASIC Magazine，12月号，171-174pp.

PC-E500/E550/1480U/1490U
▶401T
敵をよけながら時間内に霊魂を捕まえるというゲーム。——オッさん23！こと麻楊汗，ポケコンジャーナル，12月号，67-69pp.
▶2 LINE GAME
リストがわずか2行というゲーム3題。——せとけん，ポケコンジャーナル，12月号，70-71pp.
▶ヌギJONG
3枚組を2セット作る「6枚マージャンゲーム」を女の子が脱ぐようにした移植版。——せとけん，ポケコンジャーナル，12月号，72-74pp.

東京都市学校2 バッドシティの快楽学
東京都市学校の2冊目は，表現活動（マスコミ，建築を含む）をしている人が書いた東京に関するエッセイで綴られる。エッセイというより，東京都市学校でなされたレクチャーをまとめたものというほうが正しい。バッド・シティというのは，東京は未来都市であるが，かつて予想された未来とは似ても似つかぬ混沌としたバッドな末来都市だ，という意味。林海象，内田春菊，コリーブ・ブレなど顔触れは多彩。 (K)
榎本了壱監修 TOTO出版刊 ☎03(595)9689 A5版 253ページ 1,700円

柔らかい機械 思考のメカニズムの追求
人間の精神はどういうメカニズムになっているか，という厄介な問題に取り組んだ本。コンピュータの進化に従って，人間の精神に対するアプローチも変わってきた。その基本にあるのが，従来のシリアル・コンピュータから新しいコネクショニスト＝ネットワークという流れである。このコネクショニスト＝ネットワークを中心に，脳や精神を語る。やや難解だが，先月紹介した「心の社会」とあわせて読むと面白いだろう。 (K)
ジェレミー・キャンベル著 中島健訳 青土社刊 ☎03(292)7829 四六版 433ページ 2,700円

QUESTION and ANSWER

X-BASICを使っていてエラーを出すとエラーの内容とその行を出力してくれるので便利なのですが，そのあとでLISTなどとして画面を消してしまうとその行がどこであったのかがわからなくなってしまうんですね。またRUNしてそのエラー行までたどり着くのに時間がかかる場合もあるのでどうにかしてRUNさせずにエラー行を知りたいのですが，なにか方法はあるのでしょうか？

千葉県　後藤　隆

エラー行をLISTさせる方法はもちろんあります。実をいうとX-BASICにはマニュアルには載っていないが使える機能というのがあるようです。ここではX-BASICのマニュアルでは説明されていないLISTの使い方についていくつか紹介しましょう。

X-BASICのマニュアルを見てみるとLISTの使い方は，

書式　[l] list [<行番号１>] [－[<行番号2>]]

省略形　[l] l.

となっています。

問題はこの行番号の部分です。まず，多くのBASICではそうなのですが，X-BASICでも行番号を表現するのに，

.（ピリオド）

を使うことができるのです。この“.”は現在の行を表すものです。

たとえばプログラムをRUNさせて100行でブレイクキーを押して止めたのであれば“.”は100を表しますし，LOADでプログラムを読み終わったのであれば“.”はプログラムの一番最後の行を表します。

したがってプログラム実行中にエラーが出た場合も“.”は中断した行を表しているわけですから，これを使って，エラー行を表示させたいのであれば，

l..（list .）

でエラー行を表示させることができます。またエラー行から後の表示やエラー行までの表示もBASICの書式に従って，

l..－

l.－.

とすればできます。

また，もうひとつLISTコマンドの隠された機能に“ある関数の始まる行からのリストをとる”というのがあります。つまりfunc～endfuncの“func”のある行から表示するやり方です。これは，

l. 関数名－

で使うことができます。

たとえば，

```
1880 func drawmaze()
1890 /＊迷路を描く
  ：
2120 endfunc
```

となっているのであれば，

l. drawmaze－

で，

l. 1880－

と同じ働きをします。何度もリナンバーを繰り返しているときなどは便利に使えるでしょう。ただし，l. －関数名という使い方はできませんので注意してください。

（古村　聡）

X68000付属のワープロで半角文字の混ざった単語を登録しようとすると，範囲指定がされているのに「範囲指定をして下さい」となってしまって，単語を登録することができません。X68000では半角文字を辞書に登録することができないのでしょうか？

秋田県　小番　香苗

おっしゃるとおり，付属のワープロで半角文字を辞書に登録することはできません。私も決まって半角で書く文字なんか（たとえばX68000の68000の部分とか），いちいち手作業で半角に変換していました。

ワープロを使っている途中で単語を登録するんだったらまだしも，わざわざワープロを起動して単語を登録する人はいないと思います。だってDICM.Xがあるからです。そうなんです，DICM.Xなら半角文字だって辞書に登録することができるのです。きっと小番さんは，普段ASKをワープロでしか使っていないとか，そういった人なのでしょう。DICM.Xは登録する単語の品詞情報の数がワープロに比べて豊富なので，副詞や連体詞を登録するときはこちらを利用することをすすめます。

おっとDICM.Xを知らない人，システムディスクのBINディレクトリの中にありますから用意してください。使用方法は日本語ワードプロセッサ・辞書ユーティリティユーザーズマニュアルの，辞書ディスク保守ユーティリティに，詳しい説明がありますから参照してください。

X-BASICのRND文の乱数のシードを初期化するっつうのがよくわかりません。

宮城県　能登　大次

まずは次のプログラムを実行してみてください（X-BASIC上で）。

```
10 for i=0 to 10
20  print rnd()
30 next
```

見たとおり10個の乱数を表示するプログラムです。Okと表示されていますね。それではもう１回実行してみてください。どうです。なにか気づきませんか。そうですね，先ほどの実行結果とまったく同じ乱数が表示されていますね。実は乱数といっても出目は決まっているのです。上のプログラムは1000回実行しても，ずーっと同じ乱数しか出さないのです。これでは乱数とはいえませんよね。

これは乱数が数式で計算されているのが原因です。コンピュータで扱う乱数（疑似乱数）は一定の範囲の数字ができるだけデタラメな順番で出てくるように調整されています。その数字の並びには限界がありますので，何度も何度も乱数を使うと出てくる数字はいずれ最初と同じになってしまいます。この数列内部の並び方を変えるものが乱数系列と呼ばれるものです。

これを変更するのがRANDOMIZE文でそのとき与える引数が乱数のシード（種）と呼ばれるものです。X-BASICでは－32768≦ｉ≦32767の範囲で指定できるようになっています。シードを初期化するということは乱数の出目を変えるということなのです。

RANDOMIZE文が含まれていないプログラムは自動的に，

RANDOMIZE(－32768)

が指定されたことになるので，毎回同じ実行結果になってしまうのです。

では，毎回異なる乱数を出すにはどうしたらいいでしょうか。これはRANDOMIZE文の引数に内蔵タイマの秒や分やカレンダーを使う手法が有名です。たとえば，

```
5 RANDOMIZE(VAL(RIGHT$(TIME$,2))＊100)
```

を先ほどのプログラムに追加して同じように２回実行してみてください。実行結果が変わっているでしょう。

リスト1

```
================= sample.s =================
 1: * sample.s
 2:
 3:        .include       doscall.mac
 4:
 5:        .text
 6:        .even
 7:
 8: sample:
 9:        pea.l    mes1
10:        DOS      _PRINT
11:        addq.l   #4,sp
12:
13:        sub.l    a0,a1
14:        lea.l    $10(a0),a0
15:        move.l   a1,-(sp)
16:        move.l   a0,-(sp)
17:        DOS      _SETBLOCK          * メモリブロックの変更
18:        addq.l   #8,sp
19:
20: *************************
21: *    子プロセスの起動
22: *************************
23:        clr.l    -(sp)    * 自分の環境ポインタを指定
24:        pea.l    p1       * コマンドライン格納領域
25:        pea.l    fil      * コマンド行のポインタ
26:        move.w   #2,-(sp) * モード2
27:        DOS      _EXEC
28:        lea.l    14(sp),sp
29:        tst.l    d0
30:        bmi      error    * エラー
31:
32: *      ここからがモード0
33:
34:        clr.l    -(sp)    * 自分の環境ポインタを指定
35:        pea.l    p1       * コマンドライン格納領域
36:        pea.l    fil      * コマンド行のポインタ
37:        clr.w    -(sp)    * モード0
38:        DOS      _EXEC    * 子プロセスの起動
39:        lea.l    14(sp),sp
40:        tst.l    d1
41:        bmi      error
42: *************************
43: *    子プロセスの終わり
44: *************************
45:
46:        pea.l    mes2
47:        DOS      _PRINT
48:        addq.l   #4,sp
49:        DOS      _EXIT    * プロセス終了
50:
51: error:
52:        pea.l    er_mes
53:        DOS      _PRINT   * エラー表示
54:        addq.l   #4,sp
55:        DOS      _EXIT    * 異常終了
56:
57:        .data
58:
59: mes1:
60:        dc.b     '子プロセスを起動します',13,10,0,0
61: mes2:
62:        dc.b     '子プロセスを終了しました',13,10,0,0
63: er_mes:
64:        dc.b     '子プロセスの起動に失敗しました',13,10,0,0
65: fil:
66:        dc.b     'ED.X',0
67:        ds.b     90
68: p1:
69:        ds.b     256
70: p2:
71:        dc.b     0
72:
73:        .end
```

なに，変わっていない？　追加した行は，内蔵タイマの秒に100をかけたものをシードとしています。だからシードは（0≦i≦6000）です。実行結果が同じなのは，あなたが1秒以内に2回プログラムを実行したためにシードの値が変らなかったからです。さすがに3回実行すれば結果も変わるでしょう。これでは60種類の乱数パターンしかできないので，さらに複雑な計算式でシードを算出してやればいいでしょう。ちなみに，シードを100倍しているのはシードの値が近いと出てくる乱数も似てくる傾向があるからです。

そのほか，シードを初期化する以外に，

REPEAT:RND():UNTIL INKEY$(0)<>""

のようにして乱数を取り出す位置を変える手もあります。コンピュータを使うとどうしてもデタラメな値というものが作りにくいので，人間がキーボードを叩く時間間隔で乱数を調整しているわけです。

DOSコールを使って自分で作ったプログラムを子プロセスで起動したいのですが，うまく動きません。プログラマーズマニュアルを見ながらいろいろとやってみたのですが，それでも駄目なのです。どうやればいいのでしょうか。　鹿児島県　沢木　明夫

アセンブラから子プロセスを起動するにはDOSコール＄FF4B，EXECを使います。プログラマーズマニュアルを見ると，5つのモードがEXECに用意されています。このうち，子プロセスの起動にはモード0と，モード2の2つだけ使います。

まず最初にモード2を使います。モード2は指定した実行ファイルを環境で設定したパスを検索して，子プロセス起動のための情報をワークに設定します。ですからルートにない実行ファイルも，パスが通っていれば大丈夫です。

こうしてからモード0を使います。モード0はワークに設定された情報から実行ファイルをロードして実行まで行います。気がついたかと思いますが，自分でワークに必要な情報を設定しておけばモード2を無理に使う必要はありません。

たったこれだけの準備で子プロセスを起動できるのです。たぶん沢木さんが子プロセスを起動できなかったのはメモリブロックの変更をしてなかったからだと思います。つまり，親プロセスに最大メモリが割り当てられるため，子プロセスを読み込むメモリがまったくない状態だったのです。

最後に子プロセスとしてED.Xを起動させるプログラムを紹介します。参考にしてください。　（影山裕昭）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒108　東京都港区高輪2-19-13
NS高輪ビル
ソフトバンク株式会社出版部
「Oh! X質問箱」係

FROM READERS TO THE EDITOR

いよいよ新しい年，1991年がやってきました。皆さん，去年はどんな年だったでしょうか。よい年だった人たちは，今年もよい年でありますように。また，あまりよい年ではなかった人たちは，今年こそよい年でありますように。

◆11月号は発売日の昼休みに近くの本屋で買いました。テスト期間中ということもあって，みんながノートのコピーをやっているようだったけど，気のせいでしょう。教室で何気なく読んでいて，「理科系のGAME REVIEW」のところをめくったとき，近くにいた友達が「難しい本読んでるねー」と言いました。それを聞いて「あっそうか，ぼくってえらいのか」と思わず心の中でつぶやいてしまいました。3月頃に浪人にクラスチェンジしてしまいそうでこわいです。

白井　五三雄(18)愛知県

「理科系のGAME REVIEW」を読んだんだから，数学と理科は大丈夫。てなぐあいにはいきませんよね。

◆現在，私と友達の伸二君とでシミュレーションゲームを作っています。三国志に似た，“三太極”というゲームです。伸二君は絵がとても上手で，主に登場キャラクターの顔を書き，私はゲームのシステムを考えてプログラムを作ります。私たちの理想はとても高いので，なかなか紙上の段階から抜け出せません。どんなゲームかと申しますと，地形はうちの近所(半径5kmぐらい）でありまして，武将として登場するのはみんなうちの中学校の卒業生です。作業を開始してから，はや1年が過ぎましたが，これはとても笑えます。完成はいつになるか，とても予測はできませんが，うちわでそうやって騒ぐのもとても面白いものであります。

懸川　誠一(17)群馬県

知っている人しか面白くないだろうけど，それでもいいですよね。作るのを楽しんでいるんだから。

◆X68000を買う前，ほかのゲーム雑誌を見ると「ああ，X68000ってなんでもできるんだなあ」と思い，Oh!Xを読むと，「ああ，X68000ってなんにもできないんだなあ」(ちょっとオーバー)と思った。そして，1カ月半前，X68000を買ってわかった。X68000はなんでもできる。できないのは自分だ，と。でも，ずっと夢に見ていたX68000を手に入れられたのだから，すごくうれしい。

藤田　真人(17)静岡県

早くあなたのX68000がなんでもできるようになるといいですね。

◆みんな若いんだもの，なにかためらってしまうよ。受験なんて言葉，何年ぶりなんだろう。2,3カ月前に一度ハガキ出したんだけど，そのとき年をごまかしちゃった。ごめんネー。唯一女性だっていうことがアピール点かな。ところで，斎藤由貴の「運命の女」っていう詩集読みましたか？　彼女って紙一重的な表情するし，きっと天才だって思う。いま私にとって一番気になる存在なのよネ。この詩集にはその天才ぶりがあからさまに出ているのです。恥ずかしながら小説家目指している私には雷電ばりばり貫いた感じで，「ばなな」以上のショックでした。

矢吹　準子(29)福島県

紙一重的というのは，斎藤由貴さんにぴったりの言葉のような気がしますね。

◆7月，X68000を買った。10月，原付(ホンダDIO SR，140,000円)を買った。もう僕には何も残っていない。「パロディウスだ！」が出る。ほしい。金がない。悲しい。バイトしなくっちゃ。時間がなくなる。X68000が使えない。勉強もしなくっちゃ。ギターの練習もしたいし。ガォー!!　すいません。つい叫んでしまいました。

三宅　良和(21)岡山県

落ち着いてください。

◆10月21日，初めて電車に乗って高松へ行きました。以前のJR四国の車両はつまらなかったのですが，瀬戸大橋ができてからは本土からしゃれたカラフルでいろいろな列車が乗り入れてきて鉄道ファンにはうれしいかぎりです。ところで，高松へ行った理由はその筋で結果は悲惨でした。しかし，Oh!Xをしっかり読んでいれば必ず受かると自信がつきました。4月も行くぞ。

深川　哲光(32)香川県

その筋というのは，あの情報処理試験でしょうか。何種を受けたのかはわかりませんけど，高松までわざわざ行くんだから，今度こそは受かるといいですね。

◆日本シリーズが始まる。西武有利の説が多く聞こえてくるが，巨人ファンの私は大胆にも，4勝2敗で巨人の日本一を予想するのであった。もし，予想が外れたら編集部みんなで私を大笑いしてください。

成川　浩一(21)群馬県

(大笑)

◆秋の夜長，皆様いかがお過ごしでしょうか。やっぱり，ゲームにどっぷりつかっている人がほとんどでしょう。かくいう私も最近ほとんど寝不足状態が続いていて，体力が限界に。こんなとき，パソコンのように人間にも電源スイッチがついていて，パチッと切れば完全休養できたらいいと思ってしまいます。さて，ここで問題です。シャープさんはファミコンのときのようにスーパーファミコンの互換機を発売するでしょうか。答えは編集部の人が聞いてください。

岡部　誠(25)福井県

そう，スイッチはほしい。しかし，誰もスイッチを入れてくれなかったときのことを考えると恐い。タイマーをつければいいのかな？　それと，シャープさんはスーパーファミコンつきのテレビを発売するようですね。

◆この間，おまけディスクの中から「ナイトアームズのエンディングテーマ」を引っ張り出して聴いてみようかと思って，いろいろいじっていたら，なんとシステムディスクが壊れてしま

▲佐々木　哲也　京都府
後ろにあるデフォルメされたバイクの全体像も見たかったなあ。冬は寒いけど，やっぱりバイクはやめられない？

▲木下　義崇　佐賀県
とってもきれいだけど，目線が冷たくて恐いよー。「クールビューティ」という言葉もありますけどね。でも，実は男だそうです。

った。やっぱり慣れないことはするもんじゃない（実は最近ようやくX-BASICを起動させました）。P.S. 6月号の悪魔城伝説のコンフィギュレーションファイルの使い方がわからない。ド素人なもので……，情けない。

引野　奈保美(20)大阪府

悪魔城伝説のコンフィギュレーションファイルはリスト4をエディタなどで打ち込み，「AKUMA.CNF」というファイル名でセーブします。そのあとは記事中の例のようにすれば大丈夫です。

◆先日，某パソコンショップへひやかしに行くとX68000SUPER-HDの本体が透明なヤツを見つけました。昔，ファミコンのソフトで「スケルトンカセット」なるものがありましたが，ちょうどそのようなものです。「なるほど，本体の中はこうなっているのか」などと感心してしまいました。あれは市販されているのでしょうか。

倉知　和弘(15)北海道

市販はされてないでしょうね。どうせなら檜造りのやつとかがほしいな。

◆つい最近，うちの学校で文化祭をやった。それに出品するためにZ80で制御するF1のプログラムや，対戦テトリスX68000版を作ったりした。だが，しょせんプログラマは孤独なもので，F1のボディや対戦テトリスのゲーム自体にはよろこんでくれるが，プログラムがすごいとほめてくれる人はいなかった……。プログラムなんてあってもないに等しいからね。しょせん，こんなものよ。

村越　美広(18)福島県

でも，いきなり知らない人がやってきてプログラムを読みだし，「うーん，これはすごい」とか言われても結構恐いものがありますけどね。

◆家事，育児で時間のない私は，いまごろ「殺意の接吻」をやって，最後のほうで行き詰まって苦しんでいる。みんな，そんなの忘れてしまって助けてくれないだろうなあ……。ふ，古すぎる私。

吉田　早苗(28)岐阜県

子供が育つまで待って，その子にゲームを解くのを手伝ってもらうというのはどうでしょうか。

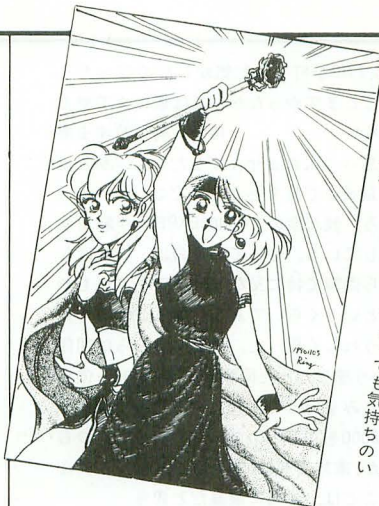
◆我が東洋大学工学部に今年，UNIXマシン（端末）が入り，後期から演習が始まった。Human68kのCOMMAND.Xしか使ったことのない（MS-DOSはちょっとだけ）僕にとって最初はわけがわからなかった。しかし，現在UNIXのすばらしさがわかり，非常に楽しいんです。まだ，機能のほんのちょっとがわかっただけですが，いつかは完全にマスターしてみせるぜ。

五島　智明(18)埼玉県

いつか，UNIXを完全にマスターして，家に1台UNIXマシンを買ってください。

◆質問したいのですが，X68000SUPER-HDにはビジュアルシェルが付属されていませんでした。手に入れる方法を教えてください。

萩原　保憲(23)神奈川県

ビジュアルシェルを手に入れる方法ですか？　それはですね……，今月号を買って

▲小川　裕美　山口県
みんなよろこべ，15歳の女の子からの投稿イラストだぞー。いやあ，明るくてとっても気持ちのいい絵だなあ。

▲岩瀬　貴代美　福岡県
この人は毎月のように載っていますが，別に女の子だからといってひいきしているわけではありません（実は多少はしているけど，いいでしょ）。

よかったでしょう。

◆11月号は職員旅行出発の日に届いたので旅行へ持っていく。今回の旅行の狙いはできたての高級旅館を経験することで，一泊2万数千円だそうだ。雲仙の東洋館の別館で東園という。ロビーに着くと琴の生演奏をやっていた。全員に昆布茶が出た。木は全部無ブシの檜だそうで，琴の後ろにあるビョウブは数千万円だそうな。この壺はン百万円で，etc……，室は50坪ほどありまして……。しかし，酒をいつものように飲みすぎる者には全部関係ありません。5千円の宿でも同じです。

平山　謙司(40)福岡県

しかし，そんなに高価な品物が周りにあると思うと，落ち着いて酔えなかったのではないでしょうか。

◆使用済みの熱転写インクリボンを巻き戻し，再びプリンタにセットしてベタ打ちする。すると，かつて印字した文章が白ヌキ文字として甦る。ディスクを処分したから，もう安心と思っているアナタ，秘密文書を誰かに読まれているかもしれませんよ。

神生　総一(24)北海道

しまった。アレを読まれると非常にヤバイじゃないか。

◆私は某市立高校に通っている17才なのですが，その日は必須科目の柔道があるので，朝から牛乳をたらふく飲んでいきました。この日の柔道ときたら……，もうおわかりでしょう。みなさん牛乳は完全食品といわれますが，おなかに消化酵素がないため飲みすぎには十分注意しましょう。

戸谷　浩史(17)群馬県

柔道があるので，牛乳を飲んでいったというのもイマイチ意味がよくわかりませんが。骨折しないようにかな？

◆郵便配達の方に表札を出してくださいといわれた。しかし，立派な表札を作ろうとすると結構お金がかかるわけで，こうなったらワープロで打ってビニールでもかぶせて玄関に貼ろう（情けないです）と早速ピコピコ，さあて印刷だ。ガチャガチャ。なんだこりゃあ。字の真ん中あたりに白く空白の部分が。説明書には印字ヘッドの損傷か，ヘッド制御回路の故障と書いてある。トホホ。

森本　俊昭(27)千葉県

表札を買うか，プリンタを直すか。それが問題だ。

◆汚い字ですみません。ちなみに僕は小学校2年生から書道をやっています。でも，それが実生活に生かされません。どうしたらよいのでしょうか。

土井　雅大(19)宮城県

書道をやめてみる，とか。

◆あごが痛い。歯医者さんへ行ったら，あごの骨がずれて関節炎を起こしているといわれた。せっかくの秋なのに固いものが食えない。ところで，編集室で通産省の情報処理技術者試験に合格している人はどのくらいいるんですか。

本間　晃(20)愛知県

情報処理試験かあ。受けろといわれれば受けないこともないですがね。ははは。

◆Oh!Xのページの下にメッセージがありますが，どうせなら上にも横にも書いたらどうでしょうか。ゲームのちょっとしたテクニックや，役に立つけれど特集にすることもできない話を数ページにわたって書いたりしたらにぎやかになっていいと思うのですが。

宮浦　慎司(17)香川県

そんな字だらけで，耳なし芳一みたいな本はやだ。

◆えっ？　またディスクがつくんですか。それは楽しみだなあ。なんてったって，私の場合Oh!Xを読み始めたのはあのディスクつきからですもんね。それに，VS.XだけではなくCOMMAND.Xから立ち上げるようになり，Human68kが理解できるようになったのも，あのディスクのおかげです(そういう人って多いと思う)。編集部の人はどう思っておられるのか知りませんが，「付録ディスク」はユーザー（Oh!X読者）を成長させます。

小川　剛(28)和歌山県

今回の付録ディスクは誰でも簡単に使えるようにVS.Xが入っているんですが，いろいろと応用的な使い方をしていただければ，また「成長」の助けになるかもしれませんね。

◆11月号の丹氏の記事は私のいいたかったことをほとんどすべて代弁してくれたみたいで読んでいて実に楽しかった。特に“真面目にやりすぎている”というのは非常に共感できるところだ。なんかいまのパソコン上で動いているフライトシミュレータというのは“現実”と“リア

ルであること”を履き違えていると思う。“G-LOCのスピードと遊び感覚”＋“本物の3D処理”＋“ジェットマニアのセンス”。この3つで最強のフライトシミュレータが完成すると私は考えている。P.S.丹さん，G-LOCは音もいいですよ。　佐藤　宰(22)京都府

「すごい」と素直に言える最強のフライトシミュレータが早くできるようになるといいですよね。

◆このところレポートの提出が山のようにあり，X68000は仮眠中です。なにせ，1週間のうちに必ず2種類（2教科）の実験レポート提出なもので休めないんですよね。実験自体は楽なんですけど，それに対する考察が面倒だったりするわけで……，来年からはもっとつらいという話だし。こりゃ，仮眠どころか冬眠，いや卒業まで熟睡するかもしれないな。　簗瀬　信悦(19)山形県

4年間寝たきり学生というのもなかなかオツなものですよ。

◆「カ・イ・ト・ウ」できなかったOPMDが解凍できた。これもOh!Xのおかげか……。いまは勝手にアレンジバージョンとかいって，サンダークロスのステージ2のBGM「スカイウォーカー」を作っている。もしも，サンダークロスの音楽がPCM音源を使っていたら，とかいってハンドクラップやオーケストラヒットとかを鳴らしまくっています。　戸辺　靖(16)栃木県

今度の付録ディスクもいろいろ楽しいものが入っているので，どうぞ遊んでやってください。

◆いつもいつも，このとーってもきれいな表紙を描いているのはどういう人ですか？　読者がOh!X LIVE in '90みたいに送ってくるのですか。11月号の凄まじいCGにとても感動しました。やっぱり，「コンピュータはすごい」のひと言です。　服部　直幸(17)熊本県

表紙はプロの人に発注しています。目次を見てもらうとわかりますが，須藤牧人さんと塚田哲也さんという方に交代で描いていただいています。

◆11月号の編集後記のところで，またディスクを配布するというのを見て，途中まで打ち込んだZMUSIC.FNCを打ち込む気がしなくなった。しかし，ここまでやったからには最後までやるかもしれない。　角田　尚(17)埼玉県

そう思って入れませんでしたから，がんばって最後まで打ち込んでください

◆このごろ，我が家のX68000EXPERTが麻雀マシンと化している。やっているのは，僕ではなくもっぱら両親で特に父などは「ワープロもやろかな」というくらいである。僕はこの姿を見てとてもうれしく思った。当初，「わしらは関係ない」という感じだった両親が，X68000を使うことに楽しみを持ってくれているからである。別に，X68000を30万円のファミコンにするつもりはないが，家族でX68000を楽しみながら使えるということは，本当に最高だと思う。　森　哲也(21)大阪府

X68000が結ぶ，親子の絆。家内安全にもX68000をどうぞ。

◆毎月，「X-BASICプログラミング調理実習」はかかさず読んでいます。今回，カード型データベースは特に実用性が大であり，興味深く読ませていただきました。「X-BASICプログラミング調理実習」以外にはC言語講座を読んでいます。新ソフトの紹介や周辺機器の案内なども楽しみにしています。そのために毎月Oh!Xを購入しているようなものです。いや，もうひとつ忘れていました。CARD.FNC用カードゲームもいいですね。　近藤　義正(47)神奈川県

CARD2.FNCもよろしく。

◆10月20，21日に学校祭があった。先輩がX68000を持ってきて音楽を鳴らしていた。21日，先輩が第2種情報処理試験を受けに行ったので，僕がX68000をいじることになった。部でX68000を使えるのは僕と先輩ぐらいなので，みんなに感心されていたんだけど，先輩が帰ってきたとたんに，僕のやれることは何もなくなってしまった。2年，たった2年のX68000歴（十分すぎる年月だが）で，こんなに差があるなんて……。やっぱり，Cコンパイラでも買わなくてはいけないな。　西谷内　誠二(16)富山県

鬼の居ぬ間に命の洗濯？　ちょっとちがいますか。

◆祖母の家の裏の山で“まつたけ”が採れた。高さ10cmくらいなのが3本。1日目は見るだけ。2日目はにおいをかぐだけ。3日目は……。　桐本　順功(14)広島県

ひょっとして食べてしまったんですか。ああ，もったいない。

◆就職用の健康診断を受けにいったところ，肝臓が悪いから酒を飲むな，甘いものを食べるな，とんでもないことをいわれてしまった。この年で酒が飲めなくなったと思うと，非常に悲しい。　岡本　肇(24)岡山県

それはきびしい。思い切り，浴びるようにお酒が飲めるように，早く肝臓を直してください。

◆こういったアンケートハガキは出したっきりで返事は来ない。友人に手紙を出すと，返事が必ず返ってくる……，とは限らないのが私の友人たちの薄情さである。こら，大吾（Oh!Xの読者であり，私の友人），誕生日プレゼントに図書券送ったのに，礼も言わんとは何事だ！　川田　剛(18)大阪府

前略，アンケートハガキもここに載れば返事がありますよ。かしこ。

◆入社以来，通勤時間往復3時間半の道程で2年と少しがんばってきた。このたび，寮に入れることとなり，通勤時間はたったの30分になる。朝寝坊はできるし，夜更かしもOK！　なんておいしいんだろう。これでX68000に触れる時間も増えた。目が悪くなりそうで，うれしい悩みですね。　魚住　雄一(21)愛知県

通勤時間はうまくやれば有効に使えますが，やっぱりたいていは無駄に過ごしてしまいますからね。できれば短いにこしたことはないですよね。

◆学生の頃からハードをやりたいと思っていたけど，いままでやる機会がなかった。いま，連載されているハードウェア工作入門は結構わかるような気がする。最近の本はソフトの話が多いけど，ハードの話もたくさん載せてください。　濱崎　健一(27)東京都

昔のパソコン雑誌はハードの話もかなり多かったですよね。いまはなかなか自分で作ろうという人も少なくなってきてますから。でも，できるだけ載せていきたいとは思います。

◆10月19日から21日の3日間，鈴鹿にF1を見にいってきました。結果はご存じのとおり，セナ＆プロストの接触リタイアもあり，亜久里選手が見事に3位表彰台に上がりました。もう，ラスト2周のときなんて，観衆総立ちの大声援で，それはそれは大変な盛り上がりでした。結局，セナのチャンピオンも決定したのですが，どうも後味の悪さが残る結果でした。来年は日本人ドライバーによる，1-2フィニッシュを期待しつつ，また足を運ぼうと思っています。　今井　慎一(20)青森県

F1は年々人気が上がっていますね。ああ，F1のいいゲームがやりたいなあ。自分でF1のコースを走ったほうがさらに面白そうですから。

▲溝畑　知幸　兵庫県
よくありがちな絵なのですが，なんとなく載せてしまいました。バックの影がドラゴンかな？　早くエメドラが出るといいですね。

▲平　智征　神奈川県
これまた，グッと落ち着いた感じの絵ですね。きれいな人ですけど，だれかモデルにした人はいるのかな？

◆X68000がそろそろこたつの上に引っ越す季節となりました。夏の間は押入の中（といっても，ふすまを取り除きステレオ，本棚などを置いていますが）でマウスを転がしていましたが，こう寒くては30分も我慢できない。さて，X68000の引越したあとで，どこで食事をしようか……。　　藤井　時光(25)高知県

こたつはいいんですけど，ゲームをやるときなどはジョイパッドをこたつの中に入れてやらないと，手がかじかんで動かないんですよね。

◆いよいよ運の谷間へと堕ちていくようです。何をやっても裏目に出るし，物を壊すし，物をなくします。運の起伏が激しいのも，困りものですが，まあ上昇するのを待つことにします。案外今度の上昇運は1億円……かも。　　岡村　克宜(19)北海道

世の中，そううまくいけばいいんですが。あんまり極度に上昇するのもあとが恐いですから，小さな幸せで我慢しておくほうがよいのでは？

◆私は週休4日制をついに実現した。これでプログラミングにも熱中できる。　　杉森　貴之(19)愛知県

大学2年生で週休4日とはなかなか。単位を順調に取っていって，4年生ぐらいになると週休6日も可能なんですがねえ。2年生ではかなり無理があるのでは？

◆私も社会人となり，長かった研修も終わり，10月から実際の業務に就きました。一応SE（システムエンジニア）を目指すことになっておりますが，. あまりにも難しい仕事であり私にやっていけるのか不安であります。しかし，そういいつつも気にせずゲームをやっているバカな社会人1年生です。　　大西　幸博(22)兵庫県

いやいや，くよくよ悩むよりは気にせずにゲームやってたほうがいいですよ。

◆僕がOh!Xを買い始めたのはディスクつきの6月号から（友人にX68000ユーザーがいたから）。そして，DōGA CGAなどのX68000のパワーに惹かれた。バックナンバーを買い漁り，当初買おうと思っていたPC-9801をやめ，X68000を夏に買おうと思ったが見事に撃沈。しかし，この冬には買おうと思っている(予定)。1月号はディスクつきになるそうですが，ありがたい。X68000を買うだけで精一杯の僕としては始めから使えるアプリケーションとして楽しみにしています。　　石原　大稔(16)岡山県

▲岡村　直也　兵庫県
このネコ，レイトレにも向いてそうだけど，イラストを書くのも結構簡単そうですね。丸書いて，サッサッサと……。

今回のハガキはF1や日本シリーズについてのものも多かったんですが，それ以上にディスクに関するハガキもいっぱいで，みなさんのディスクへの期待がひしひしと感じられました。今回のディスクも満足していただけたでしょうか。いろいろな人の苦労がこもったディスクなので有効にご利用ください。使用した感想なども待っていますのでよろしく。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★このたび，X68000とPC-9801ユーザーのサークルを作るにあたりまして，会員を募集しています。活動内容は会誌の発行や情報交換などです。興味のある方は，62円切手同封のうえ下記の住所までご連絡を。〒799-11　愛媛県西条市禎瑞352-7　藤田康弘(22)

★古旗一浩＆EXTRAからのお知らせ。MZ-700/1500/2500，X1/turbo，X68000，MSX，Macintosh etc.のグループ，EXTRAではいつでも会員募集中です。今回はX68000のグラフィックデータディスク配布のお知らせです。ただし，マジックパレット専用のデータなのであらかじめご了承ください。お値段は500円＋寄付です。詳しいことは，皆さんお誘い合わせのうえ，下記の住所までお問い合わせください。〒811-42　福岡県遠賀郡岡垣町戸切794-3　筑紫高宏

売ります

★24ピン漢字プリンタ「CZ-8PK9」を3万円（送料込み）で。箱，付属品あり。連絡は往復ハガキで。〒183　東京都府中市多磨町1-34-7　(株)ジャムコ多磨寮　山口幸一(24)

★X68000用の2Mバイト増設RAMボード「CZ-6BE2」と，数値演算プロセッサボード「CZ-6BP1」をそれぞれ2万5千円（送料込み）で。どちらも取扱説明書あり。箱なし。連絡は往復ハガキで。〒272　千葉県市川市大野町4-3086　北川進(21)

★X68000用ビデオボード「CZ-6BV1」を1万円，X1用FM音源ボード「CZ-8BS1」を8千円（どちらも送料込み）で。ともに箱，マニュアルあり。連絡は往復ハガキで。〒510-03　三重県安芸郡河芸町上野1664-1　寺本篤司(17)

★データディスクマン「DD-1」新品同様（2カ月使用）を3万円で。箱，マニュアル，付属品あり。連絡は往復ハガキで。〒321　栃木県宇都宮市瑞穂2-3市住4-57　後藤和貴(17)

買います

★MZ-2500用増設ビデオRAMボード「MZ-1R27」(互換品可)を5千円，カラーパレットボード「MZ-1M10」を4千円，拡張ユニット「MZ-1U09」を4千円（いずれも送料込み）で。連絡は往復ハガキで。〒590-05　大阪府泉南市岡田1532-3　小林聡(18)

★MZ-2000/2200用ディスクドライブ「MZ-1F07」（付属品も含めて），または同等品(ディスクドライブ＋MZ用インタフェイス＋ケーブル＋ディスクBASIC）を送料込み2万5千円から5万円程度で。シングルドライブ可。連絡は譲っていただける機器と希望価格を明記のうえ，往復ハガキで。〒390　長野県松本市女鳥羽1-8-10　江口寿美子方　岩谷政治(19)

★X1用320Kバイト外部メモリ「CZ-8BE2」を1万円（送料込み）で。完動品，付属品つきのもの。連絡は希望価格を書いて往復ハガキで。〒198-01　東京都青梅市御岳本町203　福田強(16)

★MIDI音源「CM-32L」を3万円で。また，X68000用の2Mバイト増設RAMボード（「CZ-6BE2」など）を2万5千円（送料込み）で。連絡はハガキで。〒252　神奈川県藤沢市遠藤川2番地6-505　二宮秀一(16)

バックナンバー

★Oh!Xの1988年3月号を千円(送料込み)で買います。X1のMIDIボードの製作，およびそれに関するところが完全なら可。その他，サンプルプログラムやMIDI対応MUSIC BASICが載っている号も千円（送料込み）で買います。連絡は住所，氏名，電話番号などを明記して，必ず封書で。〒869-53　熊本県芦北郡田浦町大字田浦町531-2　浜田浩論

★Oh!MZの1986年9月号，1987年3月号を各2千円（送料込み）で買います。切り抜き不可。連絡は往復ハガキで。〒904-03　沖縄県読谷村長浜1433　知花喜人(18)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今回はこちらの都合でモニタレポートの用紙の発送が遅れてしまいモニタの方々には大変ご迷惑をかけました。お詫びいたします。今月は11月号の記事に関するレポートです。

●特集「理科系のGAME REVIEW」について。「あなたがゲームを作れない理由」というのは，いままでのゲーム特集記事にないテーマであるが，非常によいと思う。ゲーム特集というのは，悪くいえば技術とは無縁でそのゲームが下火になったらなんの役にも立たない記事になりがちであった。しかし，この記事のお陰で今回の特集が単なる娯楽に終わらず，技術的な分野への橋渡しとなりえたからである。私もあまり人のことは言えないのだが，特に最近はシステムの巨大化，ブラックボックス化，さらに恵まれた市販ソフト環境のために，ユーザーの制作意欲が失われ，あまりプログラミングをしなくなってきているので，こういった記事は積極的に載せてほしい。

泉　昭彦(20)　X1turbo mode 130, PC-E500　東京都

●ゲーム特集に関して取り上げてほしいテーマ。それは「雰囲気」ですね。よくありませんか？　欧米の秀作が移植されたとたんに「クソゲー」になりさがってしまったものが。それは「雰囲気」の移植に失敗したからだと思います。下手にグラフィックがよくなって，キャラクターデザインやグラフィックを一新してしまい，日本風のグラフィックに西洋風のゲームシステムというミスマッチを生んでしまったゲーム。ちょっと考えるだけでもいろいろ出てくる。

欧米のゲームと日本のゲームを比較してなにが違うかを考えると，やっぱりゲーム自身の「雰囲気」(あやしい雰囲気のもあるけど……）が違うと思う。音楽，グラフィック，システム全部がひとつの雰囲気にうまくまとまっていて，しかもそれが自分の好みにあっていると，やっぱりのめりこんでしまうものでしょう。雰囲気というのはやっぱり大切だと思います。

高橋　毅(19)　X68000PRO, MSX2　埼玉県

●「文科系のGAME REVIEW」というのもなんか変ですけど，うん。理科系だからこそあのような味のあるレビューが書けるのではないでしょうか。高Ⅰで類系選択をせまられているいま，文系だの理系だのというのはうんざりですけど，私は味のある文の書ける理系人になりたいと思っています。知人にむちゃくちゃ数学ができるのに「私，国語なんか全然わからんよ」という人もいますが……。

安井　百合江(16)　X68000PRO　愛知県

●私が思うに，文科系は万能選手である人(というよりなんにでも興味のある人）も多いが，理科系には案外少ないのではないでしょうか。理科系は特殊な分野です。理由がそこにあるからかどうかにまで言及できませんが，理科系には「数学」と「理科」だけしかできない人が多くいて，文科系には，“得意ではないけども”という条件つきで「数学」や「理科」もできる人が結構いるということを完全に否定できるものではないと思います。

ゲーム論で特集を組める雑誌も，そうそうないでしょうから，こういったOh!Xらしい企画には好意を持ちます。ただ，私の場合，ゲームについての基礎知識があまりに少ないものですから，少々難しすぎたきらいもありました。ゲームという，とかく軽く見られがちなものに対して深く考察を入れることは大変興味があることですし，理科系らしい理屈っぽさも一応の成功だといえると思います。

ゲーム概論，といったものから，ごく限られた一点にのみテーマをしぼったものまで，偏りもなく，バランスよくまとまっていました。また，「オレにはどうせわからせんねん」という居直りにも助けられて気楽に楽しく読ませてもらいました。

浅野　憲(19)　X68000PRO, X1turboIII, X1F model 20, MZ-80C, FM-77L2, M5Jr, PC-6001, PC-1245　大阪府

●泉さんの「吾輩はパソコンである」は清水さんや，祝さんや，西川さんのような記事ですね（なにか似ている)。それはいいとして，この記事のよかったところは，マシン自身になりきった話であり，これはシャープユーザーに多いパターンであるからだ。記事を読んでいてマシン自身になれた気分になる，かなりすごい(？)記事になっている。もう一度読み返してみようっと。

船越　直弥(18)　MZ-1500　北海道

●「THE SOFTOUCH SPECIAL」について。前半のティグナスの冒険までは特にいつもと変わってないと思う。後半の麻雀以降の記事は買う側にとってみると，とても便利。実際にショップでいつも見比べたりする必要がないし，どこが違うのかをすぐに把握できる。半年に一度ぐらいずつ，ゲームのジャンルごとの比較をやってほしい。

畑　剛志(18)　X1turbo model 10, X1turboZ II, MSX/2, JR-100　北海道

●第100回を迎えたS-OSについて。昔は「C言語は8ビットに向いていない」といわれ（いって)，SLANGが生まれた。C言語もできそうな雰囲気であるが，やはりS-OSにはSLANGであると思う。僕もS-OSでシステムは組んでいるが，残念なことに毎月すべて打ち込むといったところまではいけない。忙しすぎるのだ。しかし，S-OSのシステムはいまだにすたれてはいないと思う。このS-OSの精神はいまのコンピュータ社会にはないものがある。この開発精神こそ，本当はすべての“コンピュータで遊ぶ人”に持っていてもらいたいものである。掲載についてはこのような月1本ペースをくずさないでもらいたい。

長谷川　敦士(17)　MZ-2500, MSX2　山形県

ごめんなさいのコーナー

12月号　THE SOFTOUCH SPECIAL

P.46　イメージファイトの記事で価格が間違っていました。8,800円ではなくて，9,500円（税別）です。関係者の皆様には大変ご迷惑をかけました。

12月号　INTEGRAL X1

記事中ではリスト1が「ME.BAT」，リスト2が「MENU.X1」になっていますが，実際のリストでは逆になっています。リスト中のファイル名を参照してください。申し訳ありませんでした。

バグに関するお問い合わせは
☎03(5488)1311(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

謹賀新年PRO-68K 次はいったい？

▼２度目の付録ディスクでしたが，いかがでしたでしょうか。いわゆる体験版ではなく，自前で制作しているオリジナルフロッピーですから，どうしても原価が高くなってしまいます。どうせ高くなるのならと，そのぶん内容は盛り沢山。97ページのファイル名一覧を見ていただければその凄さがおわかりでしょう。収録したものは皆さんに見てもらいたいもの，使ってもらいたいものばかりです。まだまだ，入れたいと思いながらも，容量の都合で次回まわしになったものもあります。というわけで次回のディスクは４月号を予定しています（どうしても780円は高いという方には，やはり定期購読をお勧めします）。

さて，ディスクにはある程度の大きな規模のプログラム（ゲームやツールなど）が収録できますので，皆さんからの作品をお待ちしています。もちろん採用されたものには原稿料をお支払いいたします。ぜひともオリジナル作品を投稿して，X68000の元手ぐらいは回収しましょう。

▼待望のSX-WINDOWの資料が配布できることになりました。付録ディスクに収録された膨大な資料はソフトハウスなどの技術者向けに書かれたものを編集部でまとめたものです。そこで特集もSX-WINDOWに関する解説となりました。資料と共にご活用ください。また，ウィンドウ上でのアクセサリは今後とも付録ディスクなどで配布できるよう考えています。とりあえず，アナロク時計やカレンダーなどアクセサリを制作した方はぜひとも投稿してください。

▼村田敏幸さんの「X68000マシン語プログラミング」が単行本化され，ソフトバンクの書籍編集部から発売となりました。連載の入門編に加筆訂正を加えたものです。詳しくは169ページのお知らせをご覧ください。

▼今月もページの都合でいくつかの連載がお休みになってしまいました。「マシン語カクテルin Z80's Bar」「X68000マシン語プログラミング」はお休みです。

投稿応募要領

●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒108 東京都港区高輪2-19-13 NS高輪ビル
ソフトバンク出版部
Oh！X「㋢㋼㋮㋯」係

SHIFT・BREAK

▶国会議事堂前駅駅前（ややこしい）で，即位の礼警備のおまわりさんに声をかけられ，ついつい話がその道に……。そのおまわりさん，かつてはバンドをやっていて，スティーブ・コイが心の師だったとか。「今は叩いてないんですか？」と聞いたら，「叩いてるよ，“君が代”を」だって（笑）。先月とはうってかわって警察が好きになってしまった。（浦）

▶江戸っ子シリーズ第２弾。あろうことか私は「熱い風呂とぬるい風呂」とでは，ぬるいほうが好きなのだ。銭湯などという熱湯にはとてもじゃないが入れない。長年，私の「猫肌」は「江戸っ子でない証拠」とされてきたのだが，ひとつ反証として「べらんめぃ！」の「べ」で舌をまわせることを指摘しておこう。（亀）

▶早いもので，この編集室に私が来るようになってもう３度目の冬がやってきた。時の流れは人を変えていく。それはかつてオムライスの（で）と呼ばれた私にあってもまたそうであった。風呂ふき大根。昆布の浮く湯豆腐。熱かんきゅーっと一杯，ああ，味噌煮込みうどんのうまい季節。

（単に寒いからじゃないのか？　それは。（で））

▶春が来ました。といっても桜が咲いたのではありません。雪が降らないのもきっとそのせいではないでしょう。スキーに行く予定がある私はとっても悲しい思いをしています。板もブーツもウェアも必要がないでしょう。酒とつまみとゲームさえあれば，ほかになにもいらないスキーツアーになりそうです。ところで，よかったね○○君。（S.K.）

▶僕は冬が嫌いだ。大嫌いだ。バイクの機嫌は悪くなるし，走っていても寒い。楽しみなイベントとてない。え？　クリスマス？　なーにそれ（空しい）。さて，これを書いている今は11月下旬。が，例年になく暖かい。スキー好きの人々が青ざめているのを横目に，スキーをしたことのない僕は，暖かいことを単純に喜んでいるのだ。ああ不謹慎。（A.T.）

▶あのですね。僕は不信感を募らせつつあるのですよ，シャープさん。ま，リーダーズカップからおみこしに至る，必要以上に青少年を煽った広告には目をつぶりましょう。不快ですけど。SX-WINDOWも許します。真面目に作っているのは伝わるから。でも，XCだけはなんとかしてよ。自社製品の開発によそのC使うのって悔しいでしょ？（Mu）

▶引っ越しをした。外国人不可や水商売不可，子供不可や楽器不可というのは聞いたことがあった。しかし，事態はここまできている。「固い勤め希望」，「一部上場企業社員のみ」だとさ。ひでえ話だが，東京はこういう状態なのだ。私のような自由業勤め先ナシは信用するに値しないらしい。東京は開放されたふりをしているだけなのね。（K）

▶ファミコンは前から持っていたが，ディスクシステム専用の「ボディコンクエスト」をプレイするためにツインファミコンを買った（今さら）。しかし，ゲーム自体の操作性の悪さのため，すぐ放り出して知人に貸してしまった。すると知人は１日で解いて詳しい地図まで描いてくれた。RPGには忍耐力が必要だったことを再認識した今日この頃。（KO）

▶この間，AMIGAのソフトでも買おうかと思い，秋葉原のそのテの店に顔を出してみた。ちょうど入荷日だったらしく，ソフトがドカッと入ってきたんだけど，そこにはなんと，あの「シムアース」が山のように。残念ながら，Macintoshは持っていないので関係ない。それよりアレはまだかなあ。ずいぶん待っているんだけどねえ，モリニューさん。（A）

▶疲れたなあ，なんか眠いや。あ，身体が重くて変。手が冷たぁい。そうか，あたし死ぬんだね。思えばいっぱいいろんなことがあったなぁ。みんな，さよなら，おやすみ……。……ん？　あれ？　生きてら。なぁんだ，死んだんじゃないのか。神サマの思し召しかな。じゃ，これからは世のため人のためにこの身を捧げることにすっか。（生まれ変わったE.O.）

▶仕事のあいまに編集長がゲームをやっている。「この画面を見てください！　安易に制作したゲームに見えますか？」見えるぞ。「手抜きと妥協は一切しておりません」本当だなぁ？　マシン室に放置したAMIGAではIndianapolis 500が人気。速度と画面が凄いと思っていたら，サウンドやゲーム性はもっと凄いことがわかった。（U）

▶だから先月はあらかじめ言い訳を書いておいたのですが……。ところで，X68000の最大の魅力ってなんでしょう。私はそれぞれのユーザーがパソコンの可能性を模索できる自由さにあると思うのです。でも，メーカーさんにはもっと強烈なイメージを与えてほしいですね。ユーザーをあてにすることを表明するようなプロパガンダには寂しくなります。（T）

microOdyssey

今回のディスクは「承」のディスクだ。

前回の創刊8周年記念PRO-68Kがどうしても持っていてほしいものを集めたディスク,いまなお大きな意味を持っていることはいうまでもない。Oh!Xにとって,ある意味で「起」のディスクといえる。今回はそれを継ぐものとなる。

いや,むしろ方向性は異なる。前回のディスクを作成したときにも書いたが,そのとき選ぶこともできた「別のアプローチ」というもの。それに近い。単にプログラムの集合体ではなく,ひとつのメディアであり「場所」であるもの。それが今回のディスクだ。

個人的にSX-WINDOWはシステムとしての意味以上に「場所」としての意味が大きいと思っている。システムのパフォーマンスで計るならウィンドウ環境が非ウィンドウ環境に勝てるはずはない。しかし「場所」であることがもたらす力はもっとも強い力といえる。

昔,S-OSをシステムのパフォーマンスでとらえた人は離れていき,場所としてとらえた人たちはこれまでに見られるだけのものを築きあげた。パフォーマンスという面では,S-OSはこれ以上ないというくらい不利な位置からスタートしたシステムだ。結果的にいまでは非S-OS環境よりも総合的なパフォーマンスが向上しているといっていいだろう(ただ,X68000のS-OSエミュレータ上でのWINERより表示の遅いSX-WINDOWのノートはなんとかしてほしい……)。

問題は「場所」をどのように設定するかだ。別にOh!Xでは,通信をやっていないユーザーのためにディスクをつけているわけではない。パソコン通信のネットで出回っている公開ソフトを入手したければ,それはパソコン通信を始めるべきだ。

一度誌面に掲載されたプログラムはあまり提供しないという方針に変わりはない。入力しにくく,かつ,誰にでも持っていてもらいたい便利なもの。誌面には掲載できない大きさの投稿作品などを除けば「Something New」を追求する,それが基本方針といえる。

さて,「承」があれば「転」があるだろうと予測する人は鋭い。「結」はあまりないだろうから,「転」が完成形を兼ねるとしよう。今回が前回に対して「承」であり,同時に「起」であるなら「転」は「起承転」すべてを兼ねることになるだろう。これでサイクルは完成する。問題はこれを回す「力」。

スタッフは疲弊している。

「場所」は作られた。あとは「力」が集まるのを待つしかない。

時間とディスク容量に余裕があればもっと使いやすくできた,と思う。IBMの1.4Mフォーマットならもう1枚ディスクが増えただろう。

前回のマスターディスク作成はRAMディスクとハードディスク(これもかなり贅沢な環境だったのだが)で行われたが,今回の付録ディスク作成はずいぶん整備された環境で行われた。

さっそく光磁気ディスクには大活躍してもらった。HAL研のハンディスキャナと加工ツールには本当にお世話になった。120Mバイト分に増設されたハードディスクは強力だった。

しかし,もっとも活躍したのはシステムが更新されるたびに初期化されていった数十枚のフロッピーディスクだったかもしれない。(U)

1991年2月号1月18日(金)発売

特集1 "実験的"グラフィック入門
追加解説 Z's-EX/HASH.X
特集2 SX-WINDOWへの第一歩
1990年度GAME OF THE YEARノミネート発表
Oh!X LIVE in '91
Z-BASIC用 サザエさん劇中曲
X68000用 Misty Blueオープニング

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(233)3312
	〃	書泉ブックマートB1	03(294)0011
	〃	書泉グランデ5F	03(295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(257)2660
	八重洲	八重洲ブックセンター3F	03(281)1811
	新宿	紀伊国屋書店本店	03(354)0131
	高田馬場	未来堂書店	03(200)9185
	渋谷	大盛堂書店	03(463)0511
	池袋	リブロ池袋店	03(981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(981)0111
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコン∑上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ,必要事項を明記のうえ,郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので,大切に保管してください。なお,すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は,上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店,日本IPS(株)にお申し込みください。なお,購読料金は郵送方法,地域によって異なりますので,下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(238)0700

Oh!X 1月号
■1991年1月1日発行 定価780円(本体757円)
■発行人 孫 正義
■編集人 橋本五郎
■発売元 ソフトバンク株式会社
■出版事業部 〒108 東京都港区高輪2-19-13 NS高輪ビル
Oh!X編集部 ☎03(5488)1309
出版営業部 ☎03(5488)1360 FAX 03(5488)1364
広告センター ☎03(297)0181
■印 刷 凸版印刷株式会社

SHARP

X68000EXEクラブ

EXEおみこし活動スタート!

◉まずはEXEクラブへ◉

入会無料で3つのメリット!手続きは本体同梱の入会申込ハガキを送るだけ。

メリット1▶会員番号入りオリジナル会員証電卓がもらえます。

メリット2▶各種フェアご優待・イベント案内等、数々の特典があります。

メリット3▶X68000の活用情報が手に入る「EXEおみこし活動」に参加できます。

◉EXEおみこし活動とは?◉

いわば「X68000ユーザーの、X68000ユーザによる、X68000ユーザーのための」活動です。おみこしPRESSを通じて会員同士情報を交換し、もっと68を使いこなして盛り上がってしまおう!というワケ(モデムがなくてもできるパソコン通信のようなもの?!)なので、X68000へのラブコール、会員独自のテクニック、活用法(マニアックなものでなくても他の会員には貴重!)等あなたの68自慢をドシドシ聞かせてください。会員からのメッセージは「おみこし活動隊」が整理してコミュニケーションペーパー「おみこしPRESS」にバッチリ掲載します。

10月1日開設 **おみこし活動隊へのアクセス方法**

投稿受付/大阪市淀川区西中島1丁目9-16 新大阪ストロングビル2F
X68000EXEクラブ「おみこし活動隊」係

電話受付 **06-886-0354** FAX受付 **06-304-1539**

★受付時間……平日/昼2時~夜8時
日祝/昼12時~夜8時

◉そしてEXE会員究極の◉ 「おみこしかつぎ人」を大募集!

「かつぎ人」とは、より積極的におみこし活動に参加する人のこと。EXE会員は「かつぎ人」になることで、X68000ユーザーとしてますます充実、3つのメリットで強力にサポートされます。

《メリット1》「おみこしかつぎ人の集い」に参加できます。
シャープとEXE会員の双方向コミュニケーションの場として開設されるX68000情報交換会「おみこしかつぎ人の集い」は、シャープの68スタッフと直に意見交換ができるおいしいチャンス。91年2月より全国レベルでスタートするこの会に参加すれば、68ユーザーとしてトップレベルです。

《メリット2》「おみこしPRESS」定期送付。
お店まで足を運ばなくても「おみこしPRESS」が毎号お手元に届きます。

《メリット3》「ソフトウェア・フィールド」直送。
X68000最新ソフト・各種周辺機器が一覧できる「ソフトウェア・フィールド」を半年に1回お送りします。

おみこしかつぎ人になるには……

以下の年会費(おみこしかつぎ代)が必要です。〈個人入会〉3,000円/〈グループ入会(5人1組)〉2,500円/1人 おみこし活動スタート記念特別割引期間(91年1月末まで)に限り〈個人入会〉2,000円 〈グループ入会(5人1組)〉1,500円/1人。

★詳細は「おみこし活動隊」にお電話ください。

カードデザインコミュニケーション カードデザインコンペ作品大募集

X68000で、あなただけのとびっきりのデザインカードを…。「よし、これだ!」と思ったら、あとは応募するしかない!! ソフトは自由。点数無制限。アイデアを生かした楽しいカードデザインを募集します。A・B・C 各部門毎に優秀作品を選考、オリジナルカレンダーに掲載してプレゼントします。多数の力作をお待ちしています!

《応募要領》
◎作品分類…部門A/クリスマスカード、ニューイヤーカード 部門B/バレンタインカード、バースディカード 部門C/暑中見舞カード、サークル・趣味の会お知らせカード。◎応募資格…X68000で制作されたカードに限ります。◎応募方法…ポストカードサイズ(10×15cm)作品裏面に必要事項(作品名・部門名・名前・住所・X68000使用機種名・周辺機器ソフト名)を明記し、封筒に入れて郵便にてお送りください。◎受付期間…90年10月1~91年2月28日(消印有効)。◎賞…優秀作品賞/優秀作品掲載のオリジナルカレン…及びオリジナル表彰楯を進呈。参加賞/応募者全員に優秀作品掲載のオリジナルカ…ーを進呈。※応募作品にかかわる諸権利は主催者に帰属するものとして作品の返…しません。◎発表…オリジナルカレンダー・オリジナル表彰楯の発送をもってかえさせていただきます。

◆カードデザイン屋さんOPEN◆

ご来店の皆様にカードデザイン作りの実演を楽しんでいただく「カードデザイン屋さん」を開催します。グラフィックソフト「CANVAS PRO-68K」で自分だけのカードデザインにチャレンジしてください。ぜひお近くのEXEショップへ。

カードデザインコンペを応援する ◆カードデザインスクール開講◆

専任のインストラクターが、みっちり君のカードデザインをアシストしてくれる。これはデザインコンペに参加しようとする人間にとっては、最強の助っ人だ!このスクールに行けば、優秀作品賞はもう君のもの…かもしれない。

X68000カードデザインスクール
■埼玉県/12/22(土)
コナン販売東大宮店・店頭にて開講!
問い合わせ先/048-687-3822
■千葉県/12/22(土)
コナン販売柏店・店頭にて開講!
問い合わせ先/0471-32-8541

満開の電子(でんこ)ちゃん

作：いわいいっぺい
え：岡村祭

※よい子はマネしないでね！

荻窪 圭（東京都）

やばい。「それ何！」「あ、あの」見つかってしまった。「最近、遊んでくれないと思ったら、そんなものに凝ってたのね!!」「こんなもの、捨ててやるぅ」「それを捨てられたら私の生活が……」ガシャン。そういうわけで私は彼女との関係がうまくいくようになり、とうとう同棲まで始めてしまった。現在、私が極端な寝不足に陥っているのはいうまでもない。いやいや、あっちの話ではない。夜な夜な、ベットを抜け出し、押入に忍び込んでは、隠れて電脳倶楽部を楽しんでいるのだから。あ、背後に足音が……。

おかげさまで満10歳

X BASIC HOUSE PRO SHOP　豪華絢爛10周年特別販売

新登場!

SCSIインターフェース標準装備
X68000 SUPER
(HDD無し) CZ-604C
標準価格¥348,000

さらにお求め安くなった
CZ-606D-TN /BK/GY
ドットピッチ0.31mm
CZ-603D後継機
標準価格¥79,800
続々入荷!!

最高級

10周年限定大特価
特別セットのため台数限定です

CZ-613C-TN
CZ-613D-TN
GT-4000
KGB-X68PRK-14
合計金額¥991,000
super price ¥698,000

注目!

台数限定早いもの勝ち!
EPSON GT-4000

super price ~~¥118,000~~
※ケーブルは別途お買い求め下さい

memory 6M set

余裕の6Mバイトメモリー
EXPERTII

CZ-603C
CZ-613D
KGB-X68PRK-04
合計金額¥595,000
10周年特価¥468,000

PROII

CZ-653C
CZ-603D
PIO-6BE1A
KGB-X68PRK-04
合計金額¥516,800
10周年特価¥405,000

大容量HDD

SASI type
LOGTEC SHD-040(40M) ~~¥98,000~~

SCSI type
NEC PC-HD040L (40M) ¥94,800
ITEC FC-H100 (100M) ~~¥158,000~~

注目! その2

10周年記念台数限定スキャナーセット

SUPER HD SET
CZ-613C-TN
CZ-613D-TN
GT-4000
特価¥610,000

EXPERTII SET
CZ-603C
CZ-613D
GT-4000
特価¥488,000

PROII SET
CZ-653C
CZ-603D
GT-4000
特価¥398,000

注目! その3

136桁インクジェットプリンタ
特価¥69,800
80桁ドットインパクトプリンタ
特価¥39,800

※台数に限りがあります
※ケーブルは別途お買い求め下さい

音遊MIDI!

ローコストセット
CM-32L,SX-68M,Mu-1
BH特価¥89,800

本格音源セット
CM-64,SX-68M,Mu-1
BH特価¥139,800

新一体型スピーカーセット
CM-32L,CS-10,SX-68M,Mu-1
BH特価¥98,800

電子手帳

PA-9500 定価¥48,000
BH特価販売中!!
PA-8600 定価¥28,000
BH特価¥24,800
CE-200L 定価¥2,500
BH特価¥ 2,200
CE-300L 定価¥2,800
BH特価¥ 2,500
CYBER NOTE PRO-68K
定価~~¥19,800~~
Stationary PRO-68K
定価~~¥14,800~~

宇都宮店

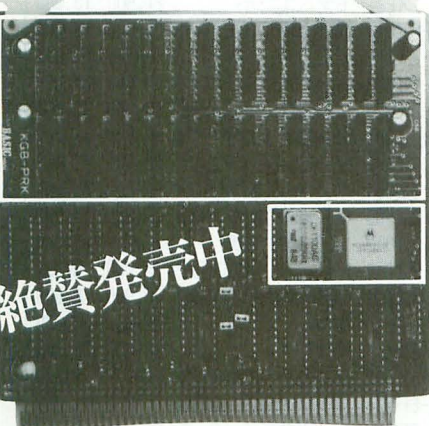
大田原店

X68000のことなら

PRO SHOP/STAFFにおまかせ下さい

価格の事から技術的な事まで何でもご相談ください
技術的なお問い合わせは登坂まで
通信販売担当は登坂、神山、片柳です。

2枚のボードが1枚になった

KGB-X68PRK

写真はKGB-X68PRK-14です

広大なメモリ空間を実現する最大4Mバイトの
高速増設メモリ

高速演算を約束してくれる
数値演算プロセッサ

- メモリアクセスノーウェイトによる高速アクセス
- CZ-6BE2、CZ-6BE4、CZ-6BP1との混在が可能
- 複数枚のKGB-X68PRKの実装が可能
- ジャンパの変更により任意のアドレス空間にメモリの配置が可能
- ジャンパの変更により数値演算プロセッサの1枚目、2枚目、未使用の選択が可能
- 1M、2M、3Mメモリモデルは購入後もメモリ増設が可能
- PRK-10、11、12、13、14にはデバイスドライバ(FLOAT3.X)が付属

製品価格一覧

型番	価格
KGB-X68PRK-00 (メモリ無し、数値演算プロセッサ無し)	¥34,000
KGB-X68PRK-01 (1Mメモリ 数値演算プロセッサ無し)	¥58,000
KGB-X68PRK-02 (2Mメモリ 数値演算プロセッサ無し)	¥74,000
KGB-X68PRK-03 (3Mメモリ 数値演算プロセッサ無し)	¥98,000
KGB-X68PRK-04 (4Mメモリ 数値演算プロセッサ無し)	¥122,000
KGB-X68PRK-11 (1Mメモリ 数値演算プロセッサ付き)	¥96,000
KGB-X68PRK-12 (2Mメモリ 数値演算プロセッサ付き)	¥112,000
KGB-X68PRK-13 (3Mメモリ 数値演算プロセッサ付き)	¥136,000
KGB-X68PRK-14 (4Mメモリ 数値演算プロセッサ付き)	¥160,000

購入後の増設費用

メモリ	
1Mバイト	¥24,000
2Mバイト	¥51,000
3Mバイト	¥76,000
数値演算プロセッサ MC68881RC16	¥38,000

※CZ-600C、601C、611C、652C、653C、662C、663Cで御使用の際にはあらかじめ専用の1Mメモリ(CZ-6BE1、A、B等)でメインメモリを2Mバイト以上にしておく必要があります。

PRK質問箱

Q、購入後のメモリ増設はどうやるのでしょう?
A、ご購入後のPRKに対するメモリの増設は半田付け等の技術を要するため原則として当社に送り返していただき増設いたします。自分でメモリ増設をする場合は通信販売のみですが必要な部品の販売も致します。御希望の方はお問い合わせ下さい。

Q、数値演算プロセッサにMC68882を使用することは可能ですか?
A、MC68882では動作しないソフトが存在するため使用できません。

Q、「数値演算プロセッサのみ」や「プロセッサ無しメモリ無し」のPRKがほしいのですが?
A、PRK-10、PRK-00の型番で商品化しております。

※最近PRKをスロットに挿入したが動作しないと言う御質問を良く受けますが、ほとんどの場合は差し込み不足が原因です。X68000のスロットは大変堅く裏蓋が閉まる状態でも差し込み不十分の場合があります。御注意ください。

おしらせ

DISK CACHER Version UP

皆様に御愛用いただいているディスクキャッシャーが高速化(従来比平均3倍)を行ないVer. UPいたしました。今回のVer. UPはハードディスクキャッシャーのみでHD-DISKCACHE Ver 2.0未満のキャッシャーを御持ちの方がVer. UPの対象となります。
御希望の方は旧バージョンのディスクのラベルと代金¥1,500(送料、税込み)を同封して現金書留で御申し込み下さい。

ビデオボードを外付けに!! ビデオボードケース(KGB-BVBX)

大好評発売中

SHARPより発売されているCZ-6BV1を外付けにする、ケースです。このケースの使用によりあなたのX68000のスロットが開放されます。

Human68k下のソフトのCRT出力を強制的に15kHZ出力にする(768×512モード除く)おまけユーティリティ付き

充実のBASIC HOUSEソフトウェア&ハードウェア

製品	価格
高速12BIT, 16CH A/Dコンバータボード(KGB-AD12) X1	¥118,000
フォトアイソレーション16BITデジタル入出力ボード(KGB-PIO) X1	¥42,000
アイソレーション16BITデジタル入出力ボード(KGB-X68PIO) X68000	¥68,000
ハンディプリンタ&インターフェース(HANDYPRINTjack) X68000	¥24,800
高速12BIT, 4CH D/Aコンバータボード(KGB-DA4) X1	¥98,000
汎用ローコストA/D&PIOボード(KGB-X1S) X1	¥19,800
高速12BIT, 16CH A/Dコンバータ(KGB-X68ADC) X68000	¥128,000
64180CPUボードMach 180(KGB-CPXB) X68000	¥98,000
ローコストMIDIインターフェース(MELODY BOX) X68000	¥16,800

製品	価格
BASIC拡張関数パッケージ(B6-6301)	¥9,800
C言語ライブラリ(B6-6305)	¥6,800
ディスクキャッシャー(B6-6304)	¥6,800
Toys & Tools (B6-6307)	¥6,800
BASIC拡張関数パッケージC言語ライブラリ付(B6-6306)	¥14,800
マイコンエディタ(B6-6303)	¥4,800
CP/M68Kエミュレータ(B6-6302)	¥19,800

全国どこでも発送可 長期クレジットOK 送料全国均一¥1,000 宅配便にて即日配送

株式会社計測技研

本社営業部/マイコンショップ/通販部 宇都宮市竹林町503―1 TEL0286-22-9811 FAX0286-25-3970
大田原営業所/マイコンショップ 大田原市美原1―13―4 TEL0287-23-5352 FAX0286-23-5364

マイコンショップ BASIC HOUSE お申し込み・お問い合せは ☎0286-22-9811(代)

SHARP

パソコン本体から周辺機器まで品数取り揃え

大特価セール実施中!!

型名	品名	正価	特価
PC-E500BL	ポケコン	28,800	**19,800**
PC-1600K	ポケコン	69,800	**49,800**
PC-1360K	ポケコン	36,800	**32,800**
PC-1360	ポケコン	29,800	**19,800**
PC-1262	ポケコン	24,800	**19,600**
PC-1248DB	ポケコン	11,000	**9,800**
PC-1280	ポケコン	24,800	**19,600**
CE-T800	ポケコンRS-232Cコンバーター	12,800	**11,800**
CE-203M	ポケコンRAM32K	32,000	**7,000**
CE-202M	ポケコンRAM16K	35,000	**6,000**
CE-201M	ポケコンRAM 8K	18,000	**3,000**
CE-1600M	ポケコンRAM32K	32,000	**16,000**
CE-1600F	ポケコンフロッピードライブ	39,800	**34,800**
CE-1600P	ポケコンプリンター	69,800	**59,800**
CE-1650F	ポケコンDISK	9,800	**8,800**
CE-161	ポケコンRAM16K	50,000	**3,800**
CE-1601M	ポケコンRAM64K	45,000	**30,000**
CE-1600E	ポケコンディスクインターフェイス	19,800	**17,800**
CE-158	ポケコンレベルコンバター	39,800	**31,300**
CE-159	ポケコンRAM 8K	35,000	**4,200**
CE-140T	ポケコンRS-232Cコンバター	9,800	**8,800**
CE-140F	ポケコンフロッピディスク	49,800	**44,800**
CE-123P	ポケコンプリンター	19,800	**17,800**
CE-120P	ポケコンプリンター	24,800	**21,800**
CE-124	ポケコンカセットインター	4,500	**4,000**
Z-VISIONplus	Z80シュミレータ デバッカー	59,800	**51,000**
UX-1	ホームコピーファクス	78,000	**63,000**
PA-9500	ハイパー電子手帳	48,000	**28,400**
CZ-300F	X13"マイクロフロッピー	79,800	**9,000**
CZ-31FS	300F増設フロッピー	59,800	**7,000**
CZ-32F	CZ-802C増設フロッピー	59,800	**6,000**
CZ-501H	X1増設用ハードディスクユニット	258,000	**60,000**
CZ-503F	CZ-830増設ドライブ	49,800	**30,000**
CZ-520F	2HD/2DDミニフロッピードライブ	118,000	**70,000**
CZ-6BG1	GPIBボード	59,800	**47,800**
CZ-6BP1	数値演算ボード	79,800	**63,800**
CZ-6BC1	FAXボード	79,800	**65,000**
CZ-6BU1	ユニバーサルI/Oボード	39,800	**33,800**
CZ-6BM1	MIDIボード	29,800	**23,800**
CZ-5BEIA	1M増設RAMボード	38,000	**19,500**
CZ-6BEIB	1M増設RAMボード	28,000	**19,500**
CZ-6BE1	1M増設RAMボード	35,000	**29,500**
CZ-6BE2	2M増設RAMボード	79,800	**特価**
CZ-6BE4	4M増設RAMボード	138,000	**特価**
CZ-6BN1	スキャナーボード	29,800	**25,300**
CZ-6BF1	RS-232C増設ボード	49,800	**42,300**
CZ-6SD1	システムラック	44,800	**38,000**
CZ-6TU	RRGBシステムチューナー	33,100	**26,500**
CZ-822C	X1G MODEL30	118,000	**39,800**
CZ-820C	X1G MODEL10	69,800	**16,800**
CZ-888C	X1TURBO	――	**特価**
CZ-8BGR2	グラフィックボードX1	14,800	**3,000**
CZ-8BF1	FDインターフェイス	14,800	**11,500**
CZ-8BK2	X1 漢字ROM	19,800	**16,800**
CZ-8BM2	232Cマウスボード	19,800	**16,800**
CZ-8BE2	320K外部メモリー	29,800	**25,300**
CZ-8BR1	立体映像セット	39,000	**33,800**
CZ-8BV2	カラーイメージボート	39,800	**32,000**
CZ-8BO1	FDインターフェイス	14,800	**8,000**
CZ-8BT2	モデムユニット	49,800	**39,800**
CZ-8EB3	拡張i/o box	33,800	**28,000**

型名	品名	正価	特価
CZ-8LM1	232cケーブル	7,200	**6,000**
CZ-8LM2	232cクロスケーブル	7,200	**6,000**
CZ-8NJ1	ジョイカード	1,700	**1,360**
CZ-8NT1	トラックボール	13,800	**11,500**
CZ-8PK10	24ドット136桁漢字プリンター	99,800	**69,000**
CZ-8PK7	24ドット80桁漢字プリンター	122,000	**59,800**
CZ-8PC4GY	24ドット熱転写カラー漢字プリンター	99,800	**59,800**
AN-S100	アンプ付スピーカー	36,600	29,500
HXD040	アイティム40Mハードディスク(ITM)	118,000	**95,000**
HXD140	40Mハードディスク内蔵用(ITM)	98,000	**75,800**
CU-14FD	カラーディスプレーアナログ0.31	74,800	**49,800**
MZ-1D10	12"モノクロディスプレー	41,800	**25,000**
MZ-1D17	15"CRT mz-5500/6500/2	124,000	**59,800**
MZ-1E05	MZ-2000 FDインターフェイス	24,500	**18,000**
MZ-1E08	プリンターI/F 2000/2200/80B	9,000	**8,000**
MZ-1E11	MZ-6500用 SFD I/F	38,000	**25,000**
MZ-1E04	MZ-2000 プリンターI/F	10,000	**6,000**
MZ-1E21	MZ-5500 GP I/F	36,000	**12,000**
MZ-1E18	MZ2000QD用インターフェイス	9,800	**3,000**
MZ-1E33	MZ6500パラレルI/F	34,800	**28,000**
MZ-1E45	MZ6500 232C I/F	50,000	**15,000**
MZ-1E32	MZ2500 パラレル I/F	30,000	**27,000**
MZ-1E44	MZ-6500 S-RN I/F	50,000	**15,000**
MZ-1E22	MZ-5500 GPIB I/F	72,800	**25,000**
MZ-1E29	RS-232Cインターフェイス 300BT	17,800	**9,800**
MZ-1E01	MZ-3500 232Cボード	28,000	**13,000**
MZ-1E14	MZ1500 QD用インターフェイス	9,800	**3,000**
MZ-1M01	MZ-2000/2200 16ビットボード	78,000	**8,000**
MZ-1M09	MZ-6500 8082-2演算プロセッサ	82,000	**30,000**
MZ-1M03	MZ-5500 数値演算	69,000	**38,500**
MZ-1M12	MZ-2861 8087 演算プロセッサ	90,000	**特価**
MZ-80P4B	136桁ドットプリンター	――	**48,000**
MZ-1P06	ドットプリンター	234,000	**45,000**
MZ-1P28	ドットプリンター漢字80桁	148,000	**118,400**
MZ-1P10A	24ドットプリンター漢字80桁	245,000	**79,000**
MZ-1P22	熱転写漢字プリンター	59,800	**25,000**
MZ-1P29	漢字プリンター136桁	168,000	**134,400**
MZ-1P30	136桁プリンター	228,000	**120,000**
MZ-1R01	MZ-2000/2200Gボード	39,800	**10,000**
MZ-1R10	MZ-5500 漢字ROM付	30,000	**9,800**
MZ-1R09	MZ-5500 V.RAM	35,000	**15,000**
MZ-1R06	MZ-5500 増設RAM	45,000	**8,000**
MZ-1R12	MZ-80B/2000/1500/700 RAM	35,000	**8,000**
MZ-1R11	MZ-5500 256KRAM	80,000	**35,000**
MZ-1R36	MZ-28611M 増設RAM	45,000	**15,000**
MZ-1R35	MZ-28611M 増設RAM	55,000	**19,000**
MZ-1R14	MZ-5500 辞書ROM	40,000	**22,000**
MZ-1R16	MZ-5500 128KRAM	30,000	**8,000**
MZ-1R27A	MZ-2500VRAM	13,000	**10,000**
MZ-1R26A	MZ-2500 増設RAM	15,000	**12,800**
MZ-1R21	漢字ROM	38,000	**13,000**
MZ-1R24	MZ-1500 辞書ROM	22,000	**6,000**
MZ-1R32	MZ-6500RAM	80,000	**40,000**
MZ-1R31	漢字ROM	28,000	**20,000**
MZ-1R28A	MZ-2500 辞書ROM	13,000	**10,000**
MZ-1R29	MZ-1P22 増設RAM17-22	32,000	**12,000**
MZ-1S13	MZ-1D17チルトスタンド	12,000	**5,000**
MZ-1T02	MZ-2200 データーレコーダー	19,800	**8,500**
MZ-1T03	MZ-5500 データーレコーダー	12,000	**8,500**
MZ-1U09	MZ-2500 拡張ボード	9,000	**7,200**
MZ-1V01	パソコン FAX	278,000	**85,000**

型名	品名	正価	特価
MZ-1X22	モデムユニット	21,800	**13,000**
MZ-2Z014	MZ-5500 TODAY	25,000	**15,000**
MZ-2Z016	MZ-5500 附属	――	**5,000**
MZ-2Z028	MZ-6500 MSDOS GWBASIC	60,000	**35,000**
MZ-2Z023	MZ-5500 GWBASIC	50,000	**30,000**
MZ-2Z031	MZ-6500 日本語ワープロ	49,800	**15,000**
MZ-2Z029	MZ-6500 TODAY	68,000	**20,000**
MZ-2Z064	MZ-6500 書院RAM付	69,800	**28,000**
MZ-2Z065	MZ-6500 書院RAMなし	49,800	**15,000**
MZ-2Z012	MZ-5500 附属	――	**5,000**
MZ-2Z013	MZ-5500 MSDOS	25,000	**20,000**
MZ-4Z001	MZ-5500 IBM変換	30,000	**8,000**
MZ-5521	本体	388,000	**55,000**
MZ-5511	本体	288,000	**35,000**
MZ-5Z013	MZ-1500 QD通信ソフト	――	**3,500**
MZ-6F03	ブランク QD DISK	450	**400**
MZ-6P18	MZ-1P18、28カットシートフィーダー	60,000	**35,000**
MZ-6P11	MZ-1P10 カットシード	95,000	**35,000**
MZ-6P29	MZ-1P29 カットシートフィーダー	50,000	**37,500**
MZ-6P27	MZ-1P27 カットシートフィーダー	58,000	**39,800**
MZ-6P06	MZ-1P06 トラクターフィード	15,000	**7,500**
MZ-6P20	MZ-1P22/17ロールホルダー	3,000	**2,700**
MZ-6Z22	M-50 CP/M86	10,000	**6,000**
MZ-6Z25	M-50 ストリーマユーティリティZプロセッサ	39,800	**15,000**
MZ-80T20A	MZ-80マシンランゲージ	6,000	**5,000**
MZ-80TUB	MZ-80 バックアップ	20,000	**8,000**
MZ-80TU	MZ-80 システムプログラム	20,000	**8,000**
MZ-80T40A	MZ-80 PASCAL	10,000	**5,000**
MZ-80T70A	MZ-80 FDOS	20,000	**7,000**
MZ-8BGK	MZ-80 BGRAM2	39,000	**10,000**
MZ-8B104	MZ200/2200 GP IBインターフェイス	45,000	**18,000**
MZ-8BG	MZ-80BGRAM1	39,000	**10,000**
MZ-8BC01	MZ200/2200 GP IBケーブル	18,000	**8,000**
AX386-FH8	MZ-8707A	1,280,000	**特価**
AX386S-F	MZ-8702B	590,000	**特価**
AX386-F	MZ-8702A	860,000	**特価**
AX386S-FH4	MZ-8706B	780,000	**特価**
AX386-FH4	MZ-8706A	1,100,000	**特価**
AX286L-FH3	MZ-8353A	598,000	**特価**
AX286L-F	MZ-8352A	428,000	**特価**
AX286D-F	MZ-8302A	278,000	**特価**
AX286D-FH4	MZ-8306A	458,000	**特価**
UE-1U01	X286L スロットBOX	5,000	**4,000**
UE-1R02	4M RAMボード	300,000	**特価**
UE-1R06	書ROMボード	32,800	**25,600**
UE-1R01	2M RAMボード	160,000	**特価**
UE-1R05	張グラフィックボード	92,000	**55,000**
UE-1R03	1M RAMボード	100,000	**特価**
UE-1R04	2M RAMボード	180,000	**特価**
UE-1P03	80桁漢字プリンタ	――	**特価**
UE-1P04	136桁漢字プリンタ	――	**特価**
UE-1P05	136桁漢字水平プリンタ	――	**特価**
UE-1P02	高速136桁漢字プリンタ	550,000	**特価**
UE-1P01	136桁漢字プリンタ	268,000	**特価**
UE-1E04	S-RNインターフェイスカード	70,000	**特価**
UE-1E02	AX286L ICカードI	45,000	**特価**
UE-1E03	5"FDインターフェイスカード	28,000	**特価**
UE-1D03	15インチカラーディスプレイ	123,000	**特価**
UE-1D02	14インチカラーディスプレイ	158,000	**特価**

ポケコン関係周辺機器サプライ製品及シャープ関係のソフトウエア全種取扱います。

FM TOWNS/FM NOTE/東芝ダイナブック、周辺機器も取扱っております。

年末謝恩セール実施中!!

セール期間中につき、送料無料サービス

'91年1月末迄

ΛIBIT
アイビット電子株式会社

シャープパソコンフェア '91 1/19・20開催
〈カードデザインコンペや、カードデザインスクールなど盛りだくさん!〉

ワープロ、パソコンお買い上げの方は、ワープロ、パソコン教室が御利用になれます。初・中・上級、無料にて実施中。

PIXELA
MacIIフルカラー・イメージクリエータ
〈ピクセカラー735〉
定価 ¥128,000
新発売!入荷

SHARP
光磁気ディスクドライブ
JY-7000
新発売!入荷

SHARP X68000シリーズ対応 ハードディスク

〈ITEM〉

HXD 040 X68000
定価¥118,000⇒特価¥95,000
HXD 042 X68000 増設用
定価¥128,000⇒特価¥102,500
HXD 140 X68000 内蔵用
定価¥98,000⇒特価¥79,800
この商品は602、603、652、653のみ使用できます。

SHARP X68000 特価表示は、オープン記念特別価格にてご提供いたします。

特価表示はTELにてご確認下さい。

CZ-603C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	特価
CZ-606D	特価
CZ-611DGY	¥305,000
CZ-613D	特価

CZ-652C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	¥275,000
CZ-606D	¥260,000
CZ-612DGY	¥290,000
CZ-605D	¥290,000

CZ-612CBK(本体)
プラス(ディスプレイ)組合せ

CZ-606D	¥330,000
CZ-605DBK	¥360,000
CZ-613DBK	¥370,000
CZ-602DBK	¥345,000

CZ-604C(本体)
プラス(ディスプレイ)組合せ

CZ-604D	特価
CZ-611DGY	¥310,000
CZ-606D	特価
CZ-605D	特価

CZ-653C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	特価
CZ-606D	特価
CZ-612DGY	¥290,000
CZ-605D	特価

CZ-602C(本体)
プラス(ディスプレイ)組合せ

CZ-606D	¥270,000
CZ-613DGY	¥310,000
CZ-605DGY	¥300,000
CZ-611DGY	¥285,000

CZ-613CBK(本体)
プラス(ディスプレイ)組合せ

CZ-604D	¥410,000
CZ-605D	¥430,000
CZ-613D	¥440,000
CZ-21HD	¥450,000

CZ-623CTN(本体)
プラス(ディスプレイ)組合せ

CZ-611DGY	¥445,000
CZ-612DGY	¥460,000
CZ-613DTN	特価
CZ-21HD	特価

パソコンゲームソフト(X1、X1t対応)

タイトル	機種	特価
トンネルズ&トロールズ	X1/X1t	¥8,330
ロードウォー2000	X1t	¥8,330
イースII	X1t	¥6,630
ソーサリアン	X1t	¥8,330
ソーサリアン(ユーティリティー)	X1t	¥3,230
ソーサリアン No.1	X1t	¥3,230
ソーサリアン No.2	X1t	¥3,230
ソーサリアン No.3	X1t	¥3,230
三国志II	X1t	¥12,580
ラスト・ハルマゲドン	X1t/Z	¥6,630
ランペルール	X1t	¥8,300
ザナドゥ	X1/X1t	¥6,630
水滸伝	X1t	¥8,330
大航海時代	X1t	¥8,330
アークス	X1t	¥8,330
信長の野望(群雄伝)	X1t	¥8,330
エグザイル	X1t	¥7,480
上海	X1/X1t	¥5,525
マスターオブモンスターズ	X1t	¥6,800
ウイザードリー	X1/t	¥8,330
パワフルまあーじゃん2	X1t	¥6,630
サイオブレイド	X1t	¥7,480
ザナドゥ シナリオII	X1t	¥4,930
琥珀色の遺言	X1t	¥8,330
倉庫番	X1/t	¥5,780
INKPOT	X1/t	¥15,300
マシン言ゲーム	X1/t	¥3,660
構造化BASICのすすめ	X1/t	¥3,660
マクロアセンブラー MACRO-80	X1/t	¥17,500
信長の野望 全国版	X1/t	¥8,330
ファンタジーIII	X1/t	¥8,330
マイト・アンド・マヂック2	X1	¥8,330
ビジレス	X1t	¥40,800
デバイスモニター	X1/t	¥4,900
JETターボターミナル	X1t	¥8,330
金庫番	X1/t	¥9,000
麻雀悟空	X1	¥5,780
ダ・ビンチ(スーパーグラフィック2)	X1/X1t	¥5,780
CZ-161LF C	X1/t	¥11,700
CZ-115LF FORTRAN	X1/t	¥11,700

アイビット推奨ディスプレイ

●シャープCZ-612D
ドットピッチ0.31
特価¥79,800

●シャープCZ-830D・BK
特価¥54,800《在庫限り》

●シャープCZ-880D
特価¥59,800

●シャープCZ-602D-BK
(15型アナログTV/3モードオートスキャン)
特価¥75,000

●三菱XC-1498CII
(14型アナログ)
ドットピッチ0.28
特価¥59,800

※シャープ周辺機器(拡張、プリンター他)も常時取り扱っております。

富士通 FM R50 FM NOTE
➡特価¥190,000

SHARP AX286N-H2 All In Note
➡特価¥358,000
外付けドライブユニット3.5インチプレゼント!

NEC PC-98HA NOTE
➡特価¥155,000

TOSHIBA J-3100GS Dyna Book
➡特価¥155,000

富士通FM TOWNSお買得セット

FM TOWNS モデル1基本セット

FM TOWNS-1	¥338,000
FMT-DP531	¥ 89,000
FMT-KB101	¥ 20,000
B276A010	¥ 20,000
特選ラック	¥ 24,000
定価合計	¥491,800

特価¥198,000

FM TOWNS モデル2基本セット

FM TOWNS-2	¥398,000
FMT-DP531	¥ 89,800
FMT-KB101	¥ 20,000
B276A010	¥ 20,000
特選ラック	¥ 24,000
定価合計	¥551,800

特価¥248,000

FM TOWNS モデル1拡張セット

FM TOWNS-1	¥338,000
HM-01T	¥ 32,800
FMT-DP531	¥ 89,800
FMT-KB101	¥ 20,000
B276A010	¥ 20,000
FMT-FD301	¥ 28,000
特選ラック	¥ 24,000
定価合計	¥552,600

特価¥258,000

FM TOWNS モデル1S基本セット

FM TOWNS-1S	¥338,000
FMT-DP531	¥ 89,800
FMT-KB101	¥ 20,000
B276A010	¥ 20,000
特選ラック	¥ 24,000
定価合計	¥491,800

特価¥218,000

FM TOWNS モデル1S拡張モデル2セット

FM TOWNS-1S	¥338,000
HM-01T	¥ 32,800
FMT-DP531	¥ 89,800
FMT-KB01	¥ 20,000
B276A010	¥ 20,000
特選ラック	¥ 24,000
定価合計	¥524,600

特価¥268,000

FM TOWNS モデル2Fセット

FM TOWNS-2F	¥378,000
FMT-KB101	¥ 20,000
FMT-DP531	¥ 89,800
B-276A010	¥ 20,000
特選ラック	¥ 24,000
定価合計	¥531,800

特価¥310,000

〈TOWNSお買い上げの方〉パソコン教室が御利用できます。初・中・上級者 無料にて実施中!

〈全商品新品完全保証付〉■シャープポケコン全商品販売中。カタログ、特価表ご請求ください(〒72)

☎0426-45-3002(駅前店)−3001(本店)−3003(教室)

FAX.0426-44-6002

●営業時間/10:00〜19:00●電話受付/20:00迄可●定休日/水曜日

SHARP SUPER XEX SHOP

アイビット電子株式会社 〒192 東京都八王子市北野町560-5

上記の広告商品はすべて店頭販売もしております

全国通信販売
北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 (普)1752505

●本誌発売時には上記価格よりさらにお求めやすい価格に変更されている場合があります。●この広告の商品にはすべて送料・消費税は含まれておりません。

クリエイト特典

- ●全商品完全保証書付(メーカー保証)
- ●全国無料配達(一部離島の方は有料になります)
- ●配達日の指定OK(日曜・祭日にかかわらずお客様のご都合にあわせて配達します)
- ●どんな商品の組合せも自由自在(ご予算、用途に応じ自由自在にシステムアップできます)
- ●中古パソコン高額下取り(今お使いのパソコンをわずかな差額でグレードアップ)
- ●お支払い方法自由(低金利の均等払い、ボーナス一括払いもご利用ください)

営業時間(定休日▶渋谷店:日曜・祭日/横浜店:水曜)
AM10:00〜PM7:00

当社はX68000の販売認定店です。どんなことでも安心してご相談ください。

冬のボーナスフェア実施中!

X68000 NEW PRO II

- ●CZ-653C(本体)……………………¥285,000
- ●CZ-603D(カラーディスプレイ)……………………¥ 84,800
- ●お好きなゲームソフト1本……………………¥ 7,800
- ■定価合計……………………¥377,600

クリエイト特価

均等払い	¥ 7,680×48回	¥ 9,890×36回	¥14,370×24回
ボーナス	なし	なし	なし

X68000 EXPERTセット

5 台 限 定 !

- ●CZ-602C-GY(本体)……………………¥356,000
- ●CZ-603D-GY(カラーディスプレイ)……………………¥ 84,800
- ■定価合計……………………¥440,800▶大特価¥279,000

大特価¥279,000

均等払い	¥12,850×24回	¥ 8,870×36回	¥ 6,920×48回
ボーナス	なし	なし	なし

X68000 NEW EXPERT II

- ●CZ-603C(本体)……………………¥338,000
- ●CZ-613D(カラーディスプレイテレビ)……………………¥135,000
- ●CZ-8NJ2……………………¥ 23,800
- ●お好きなゲームソフト1本……………………¥ 9,800
- ■定価合計……………………¥506,600

クリエイト特価

均等払い	¥ 9,970×48回	¥12,840×36回	¥18,660×24回
ボーナス	なし	なし	なし

X68000 SUPER HD

- ●CZ-623C-TN(本体・キーボード・マウス)……………………¥498,000
- ●CZ-613D-TN(カラーディスプレイ)……………………¥135,000
- ●CZ-6BP1……………………¥ 79,800
- ■定価合計……………………¥712,800

クリエイト特価

均等払い	¥ 7,320×48回	¥10,100×36回	¥13,450×24回
ボーナス	¥42,000× 8回	¥50,000× 6回	¥80,000× 4回

X68000 NEW EXPERT II

ミュージシャンセット。これもTMネットワークだよ〜!

- ●CZ-603C……………………¥338,000
- ●CZ-605D……………………¥115,000
- ●MU1.B(MIDIボード&ソフト)……………………¥ 39,800
- ●CM32L……………………¥ 69,000
- ●グラナダ……………………¥ 8,800
- ●JOYカード……………………¥ 1,800
- ■定価合計……………………¥572,400▶超特価¥458,000

X68000 NEW PRO II

ゲーマーズセット。遊んで暮らせるSET!

- ●PRO II CZ653C……………………¥285,000
- ●0.31CRT CZ603D……………………¥ 84,800
- ●グラナダ……………………¥ 8,800
- ●Y'S……………………¥ 8,700
- ●ポピュラス……………………¥ 9,800
- ●スーパーハングオン……………………¥ 8,800
- ●エージャックス……………………¥ 8,800
- ●サーク……………………¥ 8,800
- ●アールタイプ……………………¥ 7,800
- ●アナログJOYSTIC XE-1AP……………………¥ 13,800
- ■定価合計……………………¥445,100▶超特価¥353,000

※本広告に掲載の全商品の価格について消費税は含まれておりません。

X68000シリーズ用 周辺機器・ソフト オール超特価!!

型番	品名	定価	ソフト名	品名	定価
CZ-6VT1	カラーイメージユニット	¥ 69,800	MUSIC PRO	MIDI版	¥ 28,800
CZ-8NS1	カラーイメージスキャナ	¥188,000	MUSIC PRO-68K	マウスを使った楽譜ワープロ	¥ 18,800
CZ-6BE1A	IMB増設RAMボード	¥ 38,000	SOUND PRO-68K	サウンドエディタ	¥ 15,800
CZ-6BE2	2MB増設RAMボード	¥ 79,800	Sampling PRO-68K	AD PCMサンプリングエディタ	¥ 17,800
CZ-6BE4	4MB増設RAMボード	¥138,000	Musicstudio PRO-68K V.1.1	MIDIマルチレコーディングソフト	¥ 28,800
CZ-8NM3	マウス・トラックボール	¥ 9,800	OS-9/X68000	マルチタスクオペレーティングシステム	¥ 29,800
BF-68PRO	高性能CRTフィルター	¥ 19,800	PRO-68K	サイバーノート	¥ 19,800
CZ-6BP1	数値演算プロセッサ・ボード	¥ 79,800	PRO-68K	ステーショナリー	¥ 14,800
CZ-8NT1	トラックボール	¥ 13,800	Ccompiler PRO-68K	ソフト開発セット	¥ 39,800
CZ-6BM1	MIDIボード	¥ 26,800	Human 68K Ver2.0	開発ツールセット	¥ 9,800
CZ-8NJ2	アナログスティック	¥ 23,800	PIO-6BE1-A	内蔵1MRAM	¥ 25,000
CZ-6TU	パソコンチューナ	¥ 33,100	PIO-6BE2-2M	2MRAM	¥ 50,000
SX-68M	MIDI I/F	¥ 19,800	PIO-6BE4-4M	4MRAM	¥ 88,000
XE-1AP	アナログジョイパッド	¥ 13,800	MU1-B	MIDI I/F+ソフト	¥ 39,800

▲上記以外ビジネスソフト、最新ゲームソフト豊富に在庫あります。※送料はご注文の際お問合せください。 ●超特価販売中!

★この表以外の組合せ、お支払い方法もご自由にできます。
★X1シリーズ用、X68000シリーズ用各社ハードディスク/プリンタ等の周辺機器を大特価にて販売しております。
電話にてお問合せください。

●渋谷店

●横浜店

オール15%〜20%OFF

パソコン専門ショップ

総合お問合せ先☎03-486-6541(代)

ソフトクリエイト 渋谷/横浜

●渋谷店☎03-486-6541(代) 〒150:東京都渋谷区渋谷1-12-7 三和渋谷ビル 振込銀行:三井銀行 渋谷宮益坂支店(普)No.5000340

●横浜店☎045-314-4777(代) 〒221:横浜市神奈川区鶴屋町2-12-8 第1建設ビル 振込銀行:三和銀行 横浜駅前支店(普)No.310852

ACCESS

X1 エミュレータ

好評発売中

定価¥9,800

X1エミュレータはX68000上でX1シリーズのアプリケーションを実行するためのソフトエミュレータです。X1のアプリケーションを完全にソフトウェアのみでエミュレートしているため、X1上での実行速度と比較して、平均3〜5倍程度おそくなりますが、X68000のマシン上に実現した仮想X1マシンを楽しめます。また、X1とX68000の相互間でファイルを転送するためのユーティリティと専用ケーブルが付属しますので、X1上で作り上げたソフトの資産をX68000上に移行することも簡単にできます。

X1エミュレータの機能

- X1エミュレータはX1に相当する機能をエミュレート。この仮想コンピュータには最大4つのドライブが仮想的に接続。
- X1エミュレータからみたドライブはHuman68kのドライブ上にあるファイルで仮想的に実現。このファイルはX1用の5" 2Dディスクのイメージをファイル転送ユーティリティでまるごと転送したもの。
- X1エミュレータで仮想的に実現したX1は仮想ドライブから起動。このため仮想ドライブ用ファイルには、X1を立ち上げるために必要なHuBASICやCP/Mなどのシステムプログラムが必要。
- X1エミュレータでは、X1の持つVRAMを含むメモリイメージやZ80CPUを仮想的にソフトウェアで実現。

ファイル転送ユーティリティ

ディスク転送

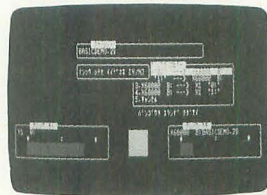
X1ディスク⟷X68000 Human68k(5" 2Dディスクイメージファイル)

- X1エミュレータではHuman68k上のディスクイメージファイルを仮想ドライブとして使用。

ファイル転送

X1 BASIC:CP/M⟷X68000 Human68k

- X1で作ったプログラム&データをX68000上で使用。

※付属の専用ケーブルをX1とX68000に接続してファイルを転送します。

X1エミュレータ Q&A

Q. ファイル転送のために別途RS-232Cケーブルを買わないといけないのですか?

A. 専用のケーブルが付属しますのでその必要はありません。

Q. X1BASICのプログラムをX68000上のX-BASICで使えますか?

A. 通常のセーブではコードが違うので使用できませんが、アスキーセーブしたファイルであればX-BASIC上でそのままロード可能です。

Q. TurboBASICで作成した住所録などの漢字を含んだデータがあるのですがX68000上にファイル転送できますか?

A. X1TurboもX68000も漢字はシフトJISコードなのでファイルの転送は可能です。ただし、漢字ROMを必要とするものはサポートしていません。

Q. Turbo用のソフトは動きますか?

A. X1用のみでTurbo専用のソフトは動きません。

Q. ゲームは動きますか?

A. 純粋にBASICでかかれたものは動きますが、プロテクトがかかったものや直接ハードをアクセスするような市販のゲームは動きません。

*タイミング等ハードウェアに依存するようなソフトは、原理上実行できない、もしくは正常に動作しない場合がありますのでご注意ください。
*一部サポートしていない機能があります。

X1エミュレータ通信販売 購入希望として住所、氏名、電話番号をお知らせください。注文書をお送り致します。

*この商品価格には消費税は含まれておりません。
*CP/Mはデジタルリサーチ社の商標です。
文中のソフトウェアは各社の商標です。
*製品の仕様、名称は予告なく変更する場合もございますのであらかじめご了承ください。

有限会社アクセス 〒101 東京都千代田区神田神保町1-64 神保町協和ビル7F
☎03(233)0200(代) FAX.03(291)7019

パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

SIG探訪PART7

ディオニュソス(ジャンプコード:SAKE)

こんな人、ぜひおいで〜

若いネットワーカーが多いなか、
ポツンと話題に乗り切れない
ら(オペ'S)のようなおじさんネットワーカー
性も含む)にも専用ボード
おじん処「赤ちょうちん」を設けています。
「背中の家族が重過ぎる。一人ぼっちになりたくて、
別れを言えば殴られる。夜更けに一人
『赤ちょうちん』…」 by COPA2
もちろん、未成年者には
喫茶「未成年」も用意しています。

ポリポリ:SIG-OP 中心に据えると存在感があります。
しう
RYo
RCR
KI-MI 「一品亭」の女将と呼ばれる一児の母
Pen
モル
LEY
長介 「赤ちょうちん」の大御所
LOT:SUB-OP 酒は下戸なので、技術担当とつっこみ専門
ぶんちゃん
MAちゃん
ワンワン OLT待合室のご隠居(二児の母)
化石人類:SUB-OP 古い考え方の部分もありますが、優しいお父さんです。
黒猫
「かみなり亭」のマスター ポリポリさんと兄弟とのうわさも(?)
BIN 「一品亭」の大御所
ゆう君

遼平(東京在住)
COPA2(仙台在住)
鹿柴 道人(東京在住)

'90.8.3「かみなり亭」にて

大規模なオフミは、年間5〜6回。主に当SIG御用達「かみなり亭」(大阪谷町6丁目)での宴です。あまりの楽しさに、はるばる東京からの参加や、ボードに出ないでもミートだけは来られる方もチラホラ……。

ほのぼのとした日だまりのような ぬくもりが心地よい。

一人ひとりがどんなにあがいても、何事もなかったかのように流れる世の中。やっぱり、自分を中心に波紋が広がってくれるような世界があってもいい──。**メインテーマがないSIGだけど、ボードに投げた小石(MSG)の波紋だけはきっちり広がってくれます。**表ボードもさることながら裏ボードが盛況で、ここを隠居部屋にしている美人妻の魅力にひかれ、アクセスするたびに未読MSGの山・山・山。**毎週土曜23時の定例OLTも大盛況**で、毎回システム終了時刻まで賑わっています。また、面倒見の良さも抜群で、**パソ通等の技術的なことはsubopのLOT氏が、その他の人生相談等はsubopの化石人類氏が、明解難解誤解(?)な回答**をさせていただいております。

その他 楽しいメニューがまだまだいっぱい!

★J&Pならではのパソコン・家電製品の会員割引もあるONLINE SHOPPING。
★J&Pだから強い!!パソコン情報をはじめとする役に立つDATA BASE。
★みんなでおしゃべりオンライントーク(CHAT機能)。
★地域別・テーマ別ボードで充実のBBS(電子掲示板)。
★ビジュアルデータもばっちり送受信できるX-MODEM。

J&P HOT LINEへのご入会はスタータキットで。

お求めは、下記のお店へ。又は現金書留にて、¥3,000+¥90(消費税3%)=¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは 〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社 J&P HOT LINE事務局宛 TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

店名	住所	電話
渋谷店	東京都渋谷区道玄坂2丁目28番4号	☎(03) 496-4141
町田店	東京都町田市森野1丁目39番16号	☎(0427)23-1313
八王子店	東京都八王子市旭町1番1号八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4−39−1	☎(0425)36-4141
本厚木店	厚木市中町3−4−3	☎(0462)25-1548
富山店	富山市桜町2−1−10	☎(0764)32-3133
金沢店	金沢市入江2−63	☎(0762)91-1130
寺地店	金沢市寺地2−3	☎(0762)47-2524
大須店	名古屋市中区大須4丁目2-48	☎(052)262-1141
㊗テクノランド	大阪市浪速区日本橋5丁目6番7号	☎(06) 634-1211
メディアランド	大阪市浪速区日本橋5丁目8番26号	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2丁目1番17号	☎(06) 634-3111
U. S. LAND	大阪市浪速区日本橋4丁目9番15号	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
梅田店	大阪市北区小松原町1−10	☎(06) 362-1141
高槻店	高槻市高槻町11番16号	☎(0726)85-1212
くずは店	枚方市楠葉花園町15番2号	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3 SENCHU PAL 2番街4F	☎(06) 834-4141
摂津富田店	高槻市大畑町24−10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4−20	☎(0720)34-1166
藤井寺店	藤井寺市岡2丁目1番33号	☎(0729)38-2111
岸和田店	岸和田市土生町2451−3	☎(0724)37-1021
さんのみや1ばん館	神戸市中央区八幡通3-2-16	☎(078)231-2111
西宮店	兵庫県西宮市河原町5−11	☎(0798)71-1171
姫路店	姫路市東延末1丁目1番住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵美須之町549	☎(075)341-3571
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4丁目4番地	☎(0734)28-1441
奈良1ばん館	奈良市三条町478−1	☎(0742)27-1111
郡山インター店	大和郡山市横田693−1	☎(07435)9-2221
熊本店	熊本市手取本町4−12	☎(096)359-7800

パーソナルコンピュータ+キーボード+マウス	CZ-888C-BK	標準価格	169,800円(税別)
14型カラーディスプレイテレビ	CZ-860D-BK	標準価格	92,200円(税別)
チルトスタンド	CZ-6ST1-B	標準価格	5,800円(税別)

クリエイティブマインドを刺激するAV機能 テレビ、ビデオ、ビデオディスクなどの映像を最大4,096色のリアルな画像で瞬時にグラフィック画面に取り込めるカラー画像デジタイズ機能を標準装備。4段階の量子化取り込み、42通りのモザイク取り込みなど多彩なトリック取り込み処理もサポート。さらにクロマキー合成、インターレーススーパーインポーズ、4,096色対応デジタルテロッパ機能、ステレオFM音源…先駆のAV機能がアートワークの領域をさらに拡げます。

AV指向の高水準ベーシックZ-BASIC搭載 多色グラフィック、カラー画像処理、ステレオFM音源、バンクメモリ対応など、ターボZシリーズが本来もつクリエイティブな機能をフルサポート。また豊富な画面モードで多色を駆使するときに便利なグラフィック用関数(HSV, RGB, HALF, CDOWN, CUP)も装備。さらにFM音源制御用ステートメントとしてX68000と命令コンパチの拡張MMLの採用によりスムーズな8音同時演奏を実現しています。

●メインメモリ128Kバイト標準装備、Z-BASICで最大576Kバイトまでサポート●1Mバイトの5インチフロッピーディスクドライブ2基搭載●JIS第1/第2水準準拠漢字、「システム・ユーザー辞書」を標準装備した高度な日本語処理機能●ニューデザインのマウス標準装備●X1ターボシリーズの豊富なソフト資産が活用できるコンパチブル設計●プリンタ、RS-232Cなど豊富なインターフェイスを装備●ドットピッチ0.39mmのハイコントラストブラウン管、15kHz/24kHzのデュアルスキャン方式採用14型カラーディスプレイテレビ(別売)

シャープ株式会社 ●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部液晶映像システム事業部第2商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161(大代表)
平成3年1月より以下におかけください ☎(03)3260-1161(大代表)

資料請求券 X1turbo ZIII Oh!X 1係

T4910217901785 雑誌 02179-1